IFIP AICT 410

Gilbert Peterson
Sujeet Shenoi
(Eds.)

Advances in Digital Forensics IX

Springer

IFIP Advances in Information and Communication Technology 410

IFIP – The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the First World Computer Congress held in Paris the previous year. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

> IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- The IFIP World Computer Congress, held every second year;
- Open conferences;
- Working conferences.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is also rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

Any national society whose primary activity is about information processing may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly, National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.

Gilbert Peterson Sujeet Shenoi (Eds.)

Advances in Digital Forensics IX

9th IFIP WG 11.9 International Conference on Digital Forensics
Orlando, FL, USA, January 28-30, 2013
Revised Selected Papers

Springer

Volume Editors

Gilbert Peterson
Air Force Institute of Technology
Wright-Patterson Air Force Base, OH 45433-7765, USA
E-mail: gilbert.peterson@afit.edu

Sujeet Shenoi
University of Tulsa
Tulsa, OK 74104-3189, USA
E-mail: sujeet@utulsa.edu

ISSN 1868-4238 e-ISSN 1868-422X
ISBN 978-3-642-41147-2 e-ISBN 978-3-642-41148-9
DOI 10.1007/978-3-642-41148-9
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013948502

CR Subject Classification (1998): K.6.5, K.4, J.1, E.3, H.3, C.2, E.5, H.2.7, F.2

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Contents

PART VIII ADVANCED FORENSIC TECHNIQUES

Contributing Authors

Irfan Ahmed is a Postdoctoral Research Associate in the Department of Computer Science at the University of New Orleans, New Orleans, Louisiana. His research interests include malware detection and analysis, digital forensics and operating systems internals.

Aswami Ariffin is a Ph.D. student in Digital Forensics at the University of South Australia, Adelaide, Australia. His research interests include digital forensics and computer security.

Nadeem Ashraf is a Security Consultant at ABM Info Tech, Islamabad, Pakistan. His research interests include digital forensics and network security.

Stefan Axelsson is a Senior Lecturer of Computer Science at the Blekinge Institute of Technology, Karlskrona, Sweden. His research interests include digital forensics, intrusion and fraud detection, visualization and digital surveillance.

Benjamin Aziz is a Senior Lecturer of Computer Security at the University of Portsmouth, Portsmouth, United Kingdom. His research interests include security and trust management, formal methods and requirements engineering, digital forensics and cloud computing.

Kamran Ali Bajwa is a Software Developer at Tabaq Software, Lahore, Pakistan. His research interests include digital forensics.

Clive Blackwell is a Research Fellow in Digital Forensics at Oxford Brookes University, Oxford, United Kingdom. His research interests include the application of software engineering and formal methods to digital forensics and information security.

Gregory Bosman was a graduate student in the Department of Computer Science, University of Pretoria, Pretoria, South Africa. His research interests include formal methods and computer security.

Frank Breitinger is a Ph.D. student in Computer Science at the University of Applied Sciences, Darmstadt, Germany; and a Researcher at the Center for Advanced Security Research Darmstadt (CASED), Darmstadt, Germany. His research interests include digital forensics, file analysis and similarity hashing.

Patrick Brennan is the President of Juola and Associates, Pittsburgh, Pennsylvania. His research interests include digital forensics and stylometry.

Joe Buchanan-Wollaston is a Research Assistant in the School of Computing Science, University of Glasgow, Glasgow, United Kingdom. His research interests include digital forensics and e-discovery.

Gary Cantrell is an Associate Professor of Criminal Justice and a Digital Forensics Examiner at the Southwest Regional Cyber Crime Institute, Dixie State University, St. George, Utah. His research interests include digital forensics, computer security and software engineering.

Vivien Chan is a Research Project Manager at the University of Hong Kong, Hong Kong, China. Her research interests include cyber criminal profiling and digital forensics.

Jim Chen is a Professor of Computer Science at George Mason University, Fairfax, Virginia. His research includes graphics, visualization, simulation and networked virtual environments.

Kim-Kwang Choo is a Senior Lecturer of Forensic Computing at the University of South Australia, Adelaide, Australia. His research interests include cyber crime and digital forensics.

Omar Choudary is a Ph.D. student in Computer Science at the University of Cambridge, Cambridge, United Kingdom. His research interests include authentication, payment protocol security, applied cryptography, hardware security and digital communications.

Kam-Pui Chow is an Associate Professor of Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include information security, digital forensics, live system forensics and digital surveillance.

David Dampier is a Professor of Computer Science and Engineering, Director of the Center for Computer Security Research, and Director of the National Forensics Training Center at Mississippi State University, Mississippi State, Mississippi. His research interests include digital forensics, information assurance and software engineering.

Stephen Esposito received his M.S. degree in Cyber Warfare from the Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio. His research interests include digital forensics and computer network operations.

Xiao-Xi Fan is a Ph.D. student in Computer Science at the University of Hong Kong, Hong Kong, China. Her research interests include digital forensics, digital profiling and data mining.

William Glisson is the Director of the Digital Forensics Laboratory and a Lecturer of Computer Forensics at the University of Glasgow, Glasgow, United Kingdom. His research interests include digital forensics and information security methods and practices in organizations.

Kalman Graffi is an Assistant Professor of Computer Science at the University of Dusseldorf, Dusseldorf, Germany. His research interests include distributed systems, social networks and security.

Rachel Greenstadt is an Assistant Professor of Computer Science at Drexel University, Philadelphia, Pennsylvania. Her research centers on the privacy and security properties of multi-agent systems and the economics of electronic privacy and information security.

Felix Grobert is an Information Security Engineer at Google in Zurich, Switzerland. His research interests include browser security, Mac OS X security and fuzz testing.

Stefan Gruner is an Associate Professor of Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include software science, formal methods and the philosophy of science.

Gaurav Gupta is an Assistant Professor of Computer Science at Indraprastha Institute of Information Technology, New Delhi, India. His research interests include digital forensics, digitized document fraud detection and mobile device forensics.

Markus Huber is a Ph.D. student in Computer Science at Vienna University of Technology, Vienna, Austria; and a Computer Security Researcher at SBA Research, Vienna, Austria. His research focuses on security and privacy issues in social networks.

Shareeful Islam is a Lecturer of Secure Software Systems at the University of East London, London, United Kingdom. His research interests include risk management, requirements engineering, security, privacy, digital forensics and cloud computing.

Salman Javaid is a Ph.D. student in Computer Science at the University of New Orleans, New Orleans, Louisiana. His research interests include digital forensics, malware analysis, security and privacy in cloud environments, and network penetration testing.

Patrick Juola is a Founder and Chief Executive Officer of Juola and Associates, Pittsburgh, Pennsylvania; and an Associate Professor of Computer Science at Duquesne University, Pittsburgh, Pennsylvania. His research interests include humanities computing, computational psycholinguistics, and digital and linguistic forensics.

Anthony Keane is the Head of Research of the Information Security and Digital Forensics Group at the Institute of Technology Blanchardstown, Dublin, Ireland. His research interests include digital forensics, cyber security, and distributed and mobile systems.

Peter Kieseberg received an M.Sc. degree in Computer Science from Vienna University of Technology, Vienna, Austria. His research interests include digital forensics, cryptography and mobile security.

Michael Kwan is an Honorary Assistant Professor of Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include digital forensics, digital evidence evaluation and the application of probabilistic models in digital forensics.

Les Labuschagne is the Executive Director of Research at the University of South Africa, Pretoria, South Africa. His research interests include project management and information security.

Pierre Lai is an Instructor of Computer Science at the University of Hong Kong, Hong Kong, China. Her research interests include cryptography, peer-to-peer networks and digital forensics.

Anyi Liu is an Assistant Professor of Computer Science at Indiana University-Purdue University Fort Wayne, Fort Wayne, Indiana. His research interests include information security and digital forensics.

Changwei Liu is a Ph.D. student in Computer Science at George Mason University, Fairfax, Virginia. Her research interests include computer security and network forensics.

Joachim Metz is an Information Security Engineer at Google in Zurich, Switzerland. His research interests include incident response and digital forensics.

Martin Mulazzani is a Ph.D. student in Computer Science at Vienna University of Technology, Vienna, Austria; and a Computer Security Researcher at SBA Research, Vienna, Austria. His research interests include privacy, digital forensics and applied security.

Sebastian Neuner is an M.Sc. student in Software Engineering and Internet Computing at Vienna University of Technology, Vienna, Austria; and a Computer Security Researcher at SBA Research, Vienna, Austria. His research interests include vulnerability discovery, penetration testing and digital forensics.

John Noecker Jr. is a Founder and Staff Scientist at Juola and Associates, Pittsburgh, Pennsylvania. His research interests include authorship attribution, author profiling and distractorless authorship verification technology.

Martin Olivier is a Professor of Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include digital forensics and privacy.

Stephen O'Shaughnessy is a Researcher in the Information Security and Digital Forensics Group at the Institute of Technology Blanchardstown, Dublin, Ireland. His research interests include digital forensics, cyber security, advanced data mining techniques and data analytics.

Gilbert Peterson, Vice Chair, IFIP Working Group 11.9 on Digital Forensics, is an Associate Professor of Computer Science at the Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio. His research interests include digital forensics and statistical machine learning.

Kaloyan Petrov is a Programmer at the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences, Sofia, Bulgaria. His research interests include parallel programming, multicore architectures and distributed computing.

Mark Pollitt, Chair, IFIP Working Group 11.9 on Digital Forensics, is an Associate Professor of Engineering Technology at Daytona State College, Daytona Beach, Florida. His research interests include digital forensics, textual and narrative theory, and knowledge management.

Antonio Pooe is a Ph.D. student in Information Systems at the University of South Africa, Pretoria, South Africa; and an Information Security Researcher at Exactech Forensics, Provo, Utah. His research interests include digital forensics, forensic readiness and information security.

Golden Richard III is a Professor of Computer Science and a University Research Professor at the University of New Orleans, New Orleans, Louisiana. His research interests include digital forensics, reverse engineering, malware analysis and operating systems internals.

Vassil Roussev is an Associate Professor of Computer Science at the University of New Orleans, New Orleans, Louisiana. His research interests are in the area of large-scale digital forensics, particularly performance, scalability, automated sampling and triage, and visual analytics support.

Michael Ryan is a Founder and Staff Scientist at Juola and Associates, Pittsburgh, Pennsylvania. His research interests include text analysis, high performance computing and systems architecture.

Ankit Sarkar is a B.Tech. student in Computer Science and Engineering at Indraprastha Institute of Information Technology, New Delhi, India. His research interests include image processing and its applications to digital forensics.

Jonathan Schluessler is a Software Development Engineer at Vector Informatik, Stuttgart, Germany. His research interests include distributed systems and information retrieval.

Sebastian Schrittwieser is a Ph.D. candidate in Computer Science at Vienna University of Technology, Vienna, Austria; and a Researcher at SBA Research, Vienna, Austria. His research interests include digital forensics, software protection and code obfuscation.

Anoop Singhal is a Senior Computer Scientist in the Computer Security Division at the National Institute of Standards and Technology, Gaithersburg, Maryland. His research interests include network security, web services security, databases and data mining systems.

Jill Slay is the Executive Dean of Information Technology at Namibia University of Science and Technology, Windhoek, Namibia; and a Professor of Forensic Computing at the University of South Australia, Adelaide, Australia. Her research interests include information assurance, digital forensics and critical infrastructure protection.

Mandhapati Venkata Srikanth is an M.Sc. student in Computer Science at the Blekinge Institute of Technology, Karlskrona, Sweden. His research interests include digital forensics.

Martin Steinebach is the Head of Media Security and IT Forensics at the Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany. His research interests include digital watermarking and robust hashing.

Ariel Stolerman is a Ph.D. student in Computer Science at Drexel University, Philadelphia, Pennsylvania. His research interests include security and privacy, applied machine learning and text analysis.

Tim Storer is a Lecturer of Software Engineering at the University of Glasgow, Glasgow, United Kingdom. His research interests include software and software-based system dependability.

Hayson Tse is a Computer Science Researcher at the University of Hong Kong, Hong Kong, China. His research interests include digital forensics, artificial intelligence and law.

Aleksandar Valjarevic is a Ph.D. student in Computer Science at the University of Pretoria, Pretoria, South Africa; and System Integration Team Leader at Vlatacom Research and Development Center, Belgrade, Serbia. His research interests include information systems security and digital forensics.

Wynand van Staden is a Senior Lecturer of Computer Science at the University of South Africa, Florida Park, South Africa. His research interests include digital forensics, anonymity and privacy.

Hein Venter is an Associate Professor of Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include digital forensics, with a current focus on the standardization of the digital forensic investigation process.

Robin Verma is a Ph.D. student in Computer Science and Engineering at Indraprastha Institute of Information Technology, New Delhi, India. His research interests include digitized document fraud detection, mobile device forensics and cloud forensics.

Harry Wechsler is a Professor of Computer Science at George Mason University, Fairfax, Virginia. His research interests include cyber security, biometrics, machine learning and data mining.

Edgar Weippl is the Research Director at SBA Research, Vienna, Austria; and an Associate Professor of Computer Science at Vienna University of Technology, Vienna, Austria. His research focuses on information security and e-learning.

Duminda Wijesekera is an Associate Professor of Information and Software Engineering at George Mason University, Fairfax, Virginia. His research interests include information, network, telecommunications and control systems security.

Christian Winter is a Research Associate in IT Forensics at the Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany. His research interests include statistical forensics and fuzzy hashing.

York Yannikos is a Research Associate in IT Forensics at the Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany. His research interests include digital forensic tool testing, synthetic test data generation and multimedia file carving.

Aleksandar Zoranic is a Ph.D. student of Computer Science at the University of New Orleans, New Orleans, Louisiana. His research interests are in the area of cloud security and live forensics via virtualization introspection, particularly memory analysis through introspection and kernel memory malware detection.

Preface

Digital forensics deals with the acquisition, preservation, examination, analysis and presentation of electronic evidence. Networked computing, wireless communications and portable electronic devices have expanded the role of digital forensics beyond traditional computer crime investigations. Practically every type of crime now involves some aspect of digital evidence; digital forensics provides the techniques and tools to articulate this evidence in legal proceedings. Digital forensics also has myriad intelligence applications; furthermore, it has a vital role in information assurance – investigations of security breaches yield valuable information that can be used to design more secure and resilient systems.

This book, *Advances in Digital Forensics IX*, is the ninth volume in the annual series produced by IFIP Working Group 11.9 on Digital Forensics, an international community of scientists, engineers and practitioners dedicated to advancing the state of the art of research and practice in digital forensics. The book presents original research results and innovative applications in digital forensics. Also, it highlights some of the major technical and legal issues related to digital evidence and electronic crime investigations.

This volume contains twenty-five edited papers from the Ninth IFIP WG 11.9 International Conference on Digital Forensics, held in Orlando, Florida, January 28–30, 2013. The papers were refereed by members of IFIP Working Group 11.9 and other internationally-recognized experts in digital forensics.

The chapters are organized into eight sections: themes and issues, forensic models, forensic techniques, filesystem forensics, network forensics, cloud forensics, forensic tools and advanced forensic techniques. The coverage of topics highlights the richness and vitality of the discipline, and offers promising avenues for future research in digital forensics.

This book is the result of the combined efforts of several individuals. In particular, we thank Mark Pollitt and Jane Pollitt for their tireless work on behalf of IFIP Working Group 11.9. We also acknowledge the support provided by the National Science Foundation, National Secu-

rity Agency, Immigration and Customs Enforcement, Internal Revenue Service and U.S. Secret Service.

Gilbert Peterson and Sujeet Shenoi

I

THEMES AND ISSUES

Chapter 1

HISTORY, HISTORIOGRAPHY AND THE HERMENEUTICS OF THE HARD DRIVE

Mark Pollitt

Abstract This paper contrasts the traditional metaphors for digital forensics – computer science, geology and archeology – with the new metaphors of history and historiography. Narratology, the study of how narratives operate, is used to develop a construct for identifying narratives from within digital evidence. Knowledge management is suggested as a core digital forensic process. The paper describes how the investigative paradigm and traditional theories of forensic science can be integrated using two theoretical constructs, the hermeneutic and narrative theories of digital forensics. Also, natural language processing techniques are used to demonstrate how subjects can be identified from the Enron email corpus.

Keywords: Forensic analysis, narratology, hermeneutics, knowledge management

1. Introduction

Digital forensics is a science that is still very much in its formative stages. Digital forensic practitioners struggle to conduct forensic examinations of ever larger data sets that are stored and communicated with increasing technical complexity by users who integrate hardware and software, such as smart phones and web applications, more and more tightly into their daily lives. Subjectively, practitioners may sense a disconnect, perhaps even frustration, between their methodologies and their goal of identifying probative evidence as digital forensic examination and analysis become more challenging.

In the following discussion, the term "digital forensic examination" refers to the documentation, identification and extraction of data from digital evidence. The term "forensic analysis" is used to describe the review and use of the data extracted from digital evidence for legal

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 3–17, 2013.

purposes. In other words, analysis contextualizes the extracted data into the operational environment.

This paper uses metaphors from the published literature to examine how the discipline of digital forensics has been traditionally viewed. Also, it investigates how these metaphors may limit the analytical aspects of the discipline. Additional metaphors are suggested to help illuminate the disconnect between examination and analysis. These metaphors suggest a paradigm that allows for the development of a theoretical construct for digital forensic examination and analysis. The theories that make up this new paradigm are suggested as a Kuhnian candidate paradigm for the science of digital forensics.

2. Traditional Forensic Science Paradigm

Inman and Rudin [10] offer one of the best known and organized paradigms for forensic science. They specify a model where there are four – and only four – forensic processes: identification, classification/individualization, association and reconstruction. The inference is that any forensic examination can be categorized into one or more of these four processes. Because, as we will shortly see, individualization is an extension of classification, we will refer to these as Inman and Rudin's four forensic questions.

Identification is simply the scientific ability to define the nature of an object. Legally, it may be sufficient to chemically identify a controlled substance such as cocaine. The possession of cocaine is prohibited; therefore, once benzoyl-methyl-ecgonine (the chemical name for cocaine) is identified in a sample, no further qualitative examination may be necessary [10].

Classification is the ability to define objects as coming from a common origin. A bullet recovered from a homicide victim, because of its dimensions, weight, shape and markings may be identified as coming from a Smith and Wesson 9 mm pistol. Because there are millions of such pistols, this is considered "class evidence." The gun that fired the bullet belongs to the "class" of all Smith and Wesson 9 mm pistols. If the gun that was used to fire this bullet is available, then the microscopic markings on the bore of the barrel could be used to determine that the recovered gun is, in fact, the source of the bullet recovered from the victim, to the exclusion of all other pistols. The result is that the evidence is "individualized."

Because of this relationship between the processes of classification and individualization, we refer to them as a single forensic question. All individualizations are not binary valued – there may be levels of

certainty. Inman and Rudin [10] give a very instructive approach for dealing with the logical problem of levels of certainty given the inability to prove a hypothesis.

The last two principles described by Inman and Rudin are association and reconstruction. Association is the ability to "infer contact" between two pieces of evidence while reconstruction is the ability to order "events in the relative space and time based on the physical evidence" [10]. A fingerprint may identify the perpetrator, but when the fingerprint is found at the scene of a crime, it associates the perpetrator with the crime scene. The entry and exit wounds on a victim's body compared with the locations of the spent bullets may allow forensic scientists to reconstruct the order of the shots and the position of the body at each impact. These two principles differ from the first two in a very significant way – they place the physical evidence in an investigative context [10].

While Inman and Rudin's forensic questions are very useful in answering questions about physical evidence (including digital evidence), they are, in many ways, too granular to contextualize much of the data in digital evidence. The investigator and prosecutor tend to use a different set of questions. They tend to use the "who," "what," "when," "where," "why" and "how" questions. While these questions are useful for investigators and prosecutors, it is difficult to directly address them using available digital forensic tools and techniques. As a result, it is necessary to find a way to bridge the gap between the two paradigms.

Most digital forensic textbooks focus on tools and artifacts. Furthermore, the textbooks provide only generalized approaches for the actual examination and analysis of digital evidence [2–4, 11]. While digital forensic practitioners need this information, it does not help them to decide what information to collect, analyze and report.

The practitioner of traditional forensic science offers, within a discipline such as firearms examination, a limited set of examinations that answer a relatively small set of questions. For example, given a fired bullet or casing, a firearms examiner can only provide information concerning the physical characteristics of the bullet and/or casing that may answer what cartridge (e.g., 9 mm or .38 Special) or firearm (e.g., Smith and Wesson revolver or Colt pistol) was used [9]. The digital forensic examiner's task is much more complex. The examiner must spend a great deal of time custom designing a process and the selection criteria that will recover, identify and extract only the most pertinent information for each individual case.

3. Engaging Other Paradigms

Farmer and Venema [8] articulate the use of a different kind of paradigm. They analogize the deleted data on a hard drive as "fossilized." They then extrapolate digital forensics to geology and archeology. They analogize the creation of data, by the operation of the computer, to the physical forces of nature such as plate tectonics and volcanoes. They call this "digital geology." What defines this aspect is that the artifacts are caused by the inherent operation of the computer and are not products of user action. In contrast, Farmer and Venema [8] use the term "digital archeology" to describe the artifacts of human intervention. These metaphorical approaches are helpful in differentiating examinations in which the activity is focused on a computer from those examinations involving user data. The use of the term archeology also implies the authorial nature of the data. The users are "writing" their history. This is not particularly helpful in identifying and selecting probative files or constructing knowledge from the data. It does, however, suggest an analogous approach, which is described below.

Farmer and Venema [8] describe the "discovery" of digital evidence as an archeology of system artifacts. This could be characterized as a "computer science" approach to digital forensics. By careful extraction of data and analysis, the examiner reconstructs computer activity. This approach has great merit in situations where the role of a computer in an investigation is as a victim, a weapon and sometimes as an instrumentality. Unfortunately, these represent the minority of digital forensic cases. Far more common are fraud, child pornography, intellectual property theft and forgery that comprise most personal and economic crimes. In such cases, a forensic examination must provide what can be gleaned from the contents of files and their fragments. Investigators and lawyers want the emails, memos, photographs, spreadsheets and social/personal connections that speak to the actions and intentions that comprise the case.

This suggests that Farmer and Venema's metaphor can be extended beyond geology and archeology to the ethnography, history, literature and sociology of the computer. Forensic practitioners are interested in searching for the activities of users and the textual (in the broadest sense) record of their activities. The user activities and content of the files are mediated by cultural, social and technical factors unique to the users. Instead of merely analyzing the content of the computer hard drive to obtain the record of a machine, it is vital to conduct the process as part archaeological dig, part anthropological study of a large, semi-organized data repository, and part unedited anthology. The mission

of the digital forensic examiner is to find the data (files and fragments) that answer the relevant questions appropriate to the particular case, characters and crime. If the fossilized records of computer activity correspond to geology and archeology, then the fossilized remains of the content might be styled as history.

4. History and Historiography

According to Collingwood [5], "[t]he value of history, then, is that it teaches us what man has done and thus what man is." In many ways, what digital forensic practitioners share with investigators is the identification and documentation of what a person has done, and by extension, what he or she is. This suggests that digital forensic practitioners are to some extent historians. Collingwood recognized that history can be viewed as both science and philosophy. He also tells us that different sciences tell us different things. In the case of history, it is the "actions of human beings that have been done in the past" and history is essentially about interpreting evidence.

In order to understand what digital forensics might gain from a historical paradigm, it is instructive to consider the activities and comments of some of the most renowned historians of our time.

David McCullough is a best-selling author, whose book, *John Adams*, about America's second President, won a Pulitzer Prize winner and was the subject of an HBO mini-series. In his 2003 Jefferson Lecture at the National Endowment of the Humanities, McCullough [13] stated: "No harm's done to history by making it something someone would want to read."

Another important historian is Ken Burns, who has truly made history accessible. Burns takes letters, photographs, film clips and interviews and weaves them into a "story." He takes the data of the historical record and turns it into a narrative.

The historian Lawrence Stone [15] defines a narrative as follows: "it is organized chronologically; it is focused on a single coherent story; it is descriptive rather than analytical; it is concerned with people not abstract circumstances; and it deals with the particular and specific rather than the collective and statistical." Indeed, Stone – and McCullough and Burns – tell us that narrative makes history accessible.

Historiography is the term for the way in which history is communicated. The selection of specific documents, photographs and historical figures mediates how history is understood. The way in which the selected historical references are presented and the way the story is told have a substantial impact on the understanding of the events that are

described. It is the historiography that makes a McCullough book or a Burns documentary powerful and compelling. The selection of the salient facts is fundamental, but their presentation ultimately defines their value.

In the same way, the selection of particular content or artifacts during a digital forensic examination is vitally important. However, if they are not presented in the context of the investigative or legal narrative, they may be overlooked or even ignored. How the content and artifacts are presented strongly influences the use of forensic data in investigations. The selection and presentation of data must create a narrative that reflects what is known about the crime, the perpetrator and the evidence. In other words, the manner in which the digital forensic examiner builds the narrative defines the hermeneutics of the hard drive.

5. Narrative as a Paradigm

Digital forensics is about several nested narratives. There is the narrative of the evidence itself: how was it created, in what order things were written and when. The evidence can be divided into the content authored by the user and the metadata that describes the provenance of the content. Farmer and Venema [8] might describe this as geological and archaeological narratives. Then there are the narratives told by the content. Because digital storage media is capable of multipurpose use, content is often related to a wide variety of user activities. In emails alone, it is common to find numerous discussions about disparate personal and professional genres involving a variety of individuals and groups. A hard drive is not like a novel; rather, it is part diary, part anthology and part notebook. While this makes the content complex, the goal of a digital forensic examination is to identify the pertinent data and present it in a useful manner. The problem can be framed as the need to select pertinent narratives from the evidence and assemble them into a coherent meta-narrative.

6. Hermeneutic Theory

Digital forensics engages two fundamental approaches: a computer science approach and an investigative approach. But a tension exists between the two approaches.

The traditional computer science forensic – or technical – approach focuses on the technical aspects of the evidence and seeks to produce reports and testimony that are scientifically defensible. In this approach, the digital forensic examiner deconstructs the physical evidence, consisting of captured and stored digital data, into a series of artifacts. The

artifacts include literal data as well as operating system, file system and network metadata. The forensic examiner seeks, through the forensic examination process, to answer the four questions described by Inman and Rudin [10] in their forensic taxonomy. Forensic examiners produce forensic reports and testimony that can be described as a meta-narrative combining the forensic process undertaken in the current examination, the operation of the technologies associated with the artifacts identified, and the artifacts themselves. These artifacts may be selected because they are probative in either the case or the legal narrative, but the examiner's forensic interest is in their presence and provenance, not in the content *per se*.

In contrast, the investigative approach seeks to discover the people and the events that constitute proof of a crime or tort. This is often accomplished using a different subset of the evidence to answer different questions. While the technical information and metadata obtained from a forensic examination may be used in investigative analysis, it is principally the content of the files and fragments that form the majority of the analysis. This does not imply that the products of the technical examination are not useful or not commonly used in the analysis. These, in large measure, form the skeleton of the analysis, but it is the content of the artifacts that provides most of the material from which the investigative narratives are created. The person with the investigative/analytical role, regardless of whether he is an attorney, investigator or forensic examiner, seeks to construct a meta-narrative that comprises two main elements, the case narrative and the legal narrative. The former is the narrative constructed by answering the investigative questions of who, what, when, where, why and how? This case narrative takes the answers to the investigative questions and instantiates them into the elements of the crime or tort, in the process creating the legal narrative. The evidence utilized to document these elements are the physical evidence (which includes the digital evidence) and the testimony (both lay and expert testimony).

Despite any notions to the contrary, these two approaches have been interrelated since the very first digital forensic examination. The relationships have changed as technology has advanced, laws have adapted and processes have evolved. In the earliest days of digital forensics, it was the investigative approach that drove the process. As practitioners gained experience, as the courts began to admit digital evidence and as the technology grew more complex, the technical approach became more important. In the mid-1990s, technology was advancing rapidly, but the ordinary citizen had little technical knowledge. To prove the reliability

Table 1. Comparison of technical and investigative approaches.

	Technical Approach	**Investigative Analysis Approach**
Evidence	Artifacts (operating system, file system, application metadata, file contents)	Content and communications from recovered artifacts
Process	Forensic examination	Investigative analysis
Context	Information technology systems	Case and legal context
Answers	Forensic questions	Investigative questions
Explicative Approach	Meta-narrative of forensic process, technology and evidentiary artifacts	Meta-narrative of case narrative and legal elements

of digital evidence, practitioners relied more and more on the technical aspects to prove the legitimacy of their proffered evidence.

Technical examinations were not, however, answering the investigative questions. Many investigators sought to perform the analysis part of the process and rely on technicians merely to provide reliable data. As digital evidence data sets grew ever larger, the ability of investigators to conduct efficient analyses diminished. The increased volume of evidence transformed the problem to one of knowledge management. The use of technologies, such as natural language processing and XML, may allow for technically-assisted knowledge management. But these technologies will not, by themselves, provide much assistance. The transformation of data into information and the transformation of information into knowledge are cognitive processes. While tools can help present data or information to analysts, they do not, on their own, produce information or knowledge. In order to transform data to information, it is necessary to examine the content and to situate it in an investigative or forensic context. As a result, it is vital to move beyond tools that focus on the technology and apply content-focused methodologies that utilize technologies to assist analysts in contextualizing data and information.

Both the technical and investigative approaches are knowledge management processes and, as such, must add value to the data. Collectively, the two approaches can be viewed as overarching theoretical constructs for digital forensics. Digital forensics cannot exist without both these approaches and, thus, the two approaches comprise the core paradigm of the science of digital forensics. Table 1 summarizes the key elements of the two approaches.

7. Hermeneutic Theory of Digital Evidence

Based on the preceding discussion, we suggest the following formal paradigm – or theory – for digital forensics:

- The legal system utilizes data stored or transmitted in digital form as evidence.
- The digital evidence must meet the reliability and authenticity tests required by the legal system.
- Forensic examinations preserve the integrity of the original evidence by technical means.
- The products of the examination process consist of artifacts, which can be sub-divided into metadata and content.
- The products of the digital forensic process are utilized to answer two interrelated sets of questions: forensic questions and the investigative questions.
- The digital forensic examination process focuses on answering the forensic questions.
- The digital forensic analysis process utilizes the products from the digital forensic examination process and generally focuses on answering the investigative questions.
- The examination and analysis phases of the digital forensic process are knowledge management processes. These processes should add value to the data and information developed during each phase.
- The forensic and investigative processing of evidence results in one or more meta-narratives, which are based on a combination of technical artifacts, metadata and content. These results form a synthesis of the forensic and investigative processes.
- The goal of the digital forensic process is to provide knowledge in the form of actionable intelligence, investigative leads, testimony and/or probative evidence.

8. Narrative Theory

The last portion of the hermeneutic theoretical construct states, in effect, that all digital evidence is – at one or more levels – a narrative. These narratives contribute to and are parts of the technical and the investigative meta-narratives. The construct also states that, in order

to create meta-narratives, content is a key element of the examination and analysis processes. Implicit in this notion is that content contains a narrative, contributes to a narrative or is in and of itself a narrative.

Since a narrative is so important to the analysis of evidence, how should we seek this crucial element? One solution is to utilize the core concepts of narratology. The field of narratology seeks to understand what constitutes a narrative, how it is structured and how it works. Bal [1], one of the pioneers of narratology, argues that there are effectively three elements that define a narrative: chronology/logic, event/action and actors.

Other researchers [6, 7, 16] suggest that it may be possible to search for narratives by utilizing an approach that attempts to identify narratives by utilizing the semantics of the content. Because this is primarily a semantic analysis, the use of sentences as a basic unit of analysis is an appropriate first approach.

The goal of a semantic analysis is to develop knowledge of subjects/actors, actions/events and chronology. Therefore, a suitable approach is to apply natural language processing to identify and recognize relationships among these elements. Since nouns and verbs are rough proxies for subjects/actors and actions/events, respectively, techniques that focus on these grammatical structures can improve the identification, efficiency and comprehension of narratives.

Beyond the identification of narratives, it is essential, given the vast quantities of digital evidence, to be able to identify particular narratives that are probative. At present, there do not seem to be technologies that can, by themselves, accurately evaluate the probative value of any given narrative. However, the ability of human analysts to evaluate data is limited by things like attention, fatigue and preconceptions about the data. It would, therefore, be useful to develop approaches that simultaneously reduce the volume of data that needs to be reviewed by analysts and enhance the ability of analysts to identify probative data. Reading a half million emails is not practical. Reducing this number and simultaneously increasing the likelihood that what is read is probative should be a goal of knowledge management in the digital forensic context.

These aspects of the analysis of digital evidence suggest the following hypothesis:

- **Hypothesis 1:** The identification of narratives by automated means will contribute to the efficiency and effectiveness of a forensic examiner or investigative analyst.

A second dependent hypothesis is:

- **Hypothesis 2:** Any automated process that improves the ability of a forensic examiner or investigative analyst to quickly identify probative narratives will improve the efficiency and effectiveness of the process.

Two other hypotheses are:

- **Hypothesis 3:** The use of nouns and verbs will assist in the identification of general and probative narratives more economically than reading complete texts.

- **Hypothesis 4:** Natural language processing software can assist in the identification of probative narratives by the use of lexical, grammatical and semantic techniques.

9. Identifying Narrative Elements

Todorov and Weinstein [16] identify the elements of a narrative as: (i) the subject, identified as a noun; (ii) the predicate, which is "always" a verb; and (iii) the adjective, which infuses a "quality" without changing the situation. After describing a grammatical analysis of structure, Todorov and Weinstein suggest that this grammatical approach can be used to further the study of narrative syntax, theme and rhetoric. If a corpus of textual digital evidence were to be processed using a "named entity" extraction technique, then it might be possible to identify the subjects of the narratives contained in the corpus.

A series of experiments were conducted to test this approach. In order to provide an investigative focus for the experiments, it was decided to study a specific and well-documented fraudulent scheme perpetrated by a group of Enron employees. The scheme involved the fraudulent raising of the price of electricity purchased by California power companies. This scheme was described by McLean and Elkind [14] in a book entitled *The Smartest Guys in the Room: The Amazing Rise and Scandalous Fall of Enron*.

In order to test the utility of natural language processing in identifying the subjects of this scheme, a subset of the Enron corpus comprising 2,385 emails from user-labeled directories involving the word "California" were processed for named entities. These emails resulted in more than 11,000 uniquely-named entities. The same natural language processing Python script was used to process Chapter 17 (*Gaming California*) in Mclean and Elkind's book [14], which discussed the fraudulent scheme.

Table 2. Results of named entity extraction.

Source	Uniquely-Named Entities
McLean and Elkind Chapter 17	151
Enron Employee California Emails (n = 2,385)	11,336
McLean and Elkind Chapter 17 and Emails	98

Table 2 shows the results of the experiment. The 98 uniquely-named entities, which were in the book chapter and in the emails, represent 64.9% of the 151 uniquely-named entities in the book chapter. In other words, a little less than 65% of the named entities found in the book chapter were also found in the emails involving the term "California." This ratio, which we call the "commonality ratio," is given by:

$$C = \frac{NI^k}{NE^k \cap NE^q}$$

where NE^k is the set of named entities in the known text and NE^q is the set of named entities located in the questioned text.

Table 3. Results of named entity extraction from Chapter 17 and emails.

	Chap. 17 Only	Chap. 17 and Emails	Commonality Ratio
Original Results	151	98	64.9%
Misclassified and Misspelled Removed	143	98	68.5%
External Literary References Removed	139	98	70.5%

A review of the results, while promising, raised some concerns about the "cleanliness" of the data. A number of the named entities had been misclassified by the software and were not actually named entities. Other words were misspelled and were incorrectly classified. Also, a number of correctly-identified named entities, primarily in the book chapter, were names of journalistic sources. In an effort to clean the data, all the misclassified and misspelled words and journalistic references were removed. Table 3 shows the results.

In an effort to determine if the approach is capable of discriminating between different sets of narratives, a second experiment was undertaken. Another chapter from McLean and Elkind's book, which does not focus on the California scheme, but on an accounting fraud masterminded by Andrew Fastow, was used. This chapter, entitled *Andy Fastow's Secrets* (Chapter 11), was processed in the same manner as Chapter 17. The

Table 4. Results of named entity extraction from Chapter 11 and emails.

	Chap. 11 Only	**Chap. 11 and Emails**	**Commonality Ratio**
Original Results	157	62	39.5%
Misclassified and Misspelled Removed	153	62	40.5%
External Literary References Removed	152	62	40.8%

data was also cleaned in the same fashion as in the previous experiment. The results are shown in Table 4.

Given the large number of emails that contained the names of people and organizations that were pertinent to a wide range of Enron operations, it is remarkable that the commonality ratios are so different. This is further reinforced by the fact that several major actors in the Enron saga are named in both chapters, thus increasing the ratio for both sets. Even in the worst case scenarios – where the minimum commonality between the California emails and Chapter 17 is compared with the maximal commonality between the California emails and Chapter 11 – the difference is 62.8%. This would appear to be a significant level of differentiation between two narratives that share a large number of common subjects.

10. Conclusions

Digital forensics is a nascent science that lacks a substantial theoretical foundation. Most of its existent theory is borrowed from computer science or forensic science. This paper has proposed two theoretical constructs, the hermeneutic and narrative theories of digital forensics, that attempt to integrate the scientific and investigative aspects of digital forensics. Narrative theory seeks to reify the fundamentally textual nature of digital evidence in the specific genre of the narrative. Proffering such theoretical constructs may be viewed as ambitious, perhaps even presumptuous, but they are offered in the spirit of research. The constructs can be evaluated, criticized, tested, disproved and improved. Any discipline that is deemed "scientific" as specified by Kuhn must have an underlying paradigm. The theoretical constructs are offered as a "shared example" or "candidate paradigm" in the sense of Kuhn [12]:

> History suggests that the road to a firm research consensus is extraordinarily arduous. In the absence of some candidate for paradigm, all the facts that could possibly pertain to the development of a given science are likely to seem equally relevant. Only very occasionally, as in the case of ancient statics, dynamics, and geometrical optics, do facts collected

> with so little guidance from pre-established theory speak with sufficient clarity to permit the emergence of a first paradigm.

The use of natural language processing to demonstrate the potential for the use of narrative shows promise. While the experiments conducted were neither elegant nor definitive, they suggest that combining narratology and natural language processing have promise and are worthy of further research.

References

[1] M. Bal, *Narratology: Introduction to the Theory of Narrative*, University of Toronto Press, Toronto, Canada, 2009.

[2] B. Carrier, *File System Forensic Analysis*, Pearson Education, Upper Saddle River, New Jersey, 2005.

[3] H. Carvey, *Windows Forensic Analysis DVD Toolkit*, Syngress, Burlington, Massachusetts, 2007.

[4] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*, Academic Press, Waltham, Massachusetts, 2011.

[5] R. Collingwood, *The Idea of History*, Oxford University Press, Oxford, United Kingdom, 1994.

[6] A. de Waal, J. Venter and E. Barnard, Applying topic modeling to forensic data, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 115–126, 2008.

[7] J. Doyle, Mapping the World of Consumption: Computational Linguistics Analysis of the Google Text Corpus, Working Paper, Cardiff Business School, Cardiff University, Cardiff, United Kingdom, 2010.

[8] D. Farmer and W. Venema, *Forensic Discovery*, Addison-Wesley, Upper Saddle River, New Jersey, 2005.

[9] E. Hueske, Firearms and tool marks, in *The Forensic Laboratory Handbook Procedures and Practice*, A. Mozayani and C. Noziglia (Eds.), Humana Press, Totowa, New Jersey, pp. 143–176, 2006.

[10] K. Inman and N. Rudin, *Principles and Practice of Criminalistics: The Profession of Forensic Science*, CRC Press, Boca Raton, Florida, 2000.

[11] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*, Pearson Education, Indianapolis, Indiana, 2002.

[12] T. Kuhn, *The Structure of Scientific Revolutions*, University of Chicago Press, Chicago, Illinois, 1996.

[13] D. McCullough, The Course of Human Events, David McCullough Lecture, National Endowment for the Humanities, Washington, DC (`www.neh.gov/about/awards/jefferson-lecture/david-mccullough-lecture`), 2003.

[14] B. McLean and P. Elkind, *The Smartest Guys in the Room: The Amazing Rise and Scandalous Fall of Enron*, Portfolio, New York, 2003.

[15] L. Stone, The revival of narrative: Reflections on a new old history, *Past and Present*, no. 85, pp. 3–24, 1979.

[16] T. Todorov and A. Weinstein, Structural analysis of narrative, *Novel: A Forum on Fiction*, vol. 3(1), pp. 70–76, 1969.

Chapter 2

PROTECTING THIRD PARTY PRIVACY IN DIGITAL FORENSIC INVESTIGATIONS

Wynand van Staden

Abstract The privacy of third parties is an important issue in digital forensic investigations. The typical steps followed during an investigation require that the equipment used during the commission of the crime be seized and analyzed in a manner that complies with accepted investigative policy and practice. The analysis of the data on the seized equipment provides the investigator, who may not necessarily be associated with a law enforcement agency, with the opportunity to access personally identifiable information of persons or entities who may not be linked to the crime; this is especially true in multi-user environments.

This paper considers the key aspects surrounding privacy protection of third parties during the *post mortem* data analysis phase of digital forensic investigations. It proposes a framework that helps protect privacy without reducing the effectiveness of an investigation. The design includes a profiling component that analyzes a search corpus and a filtering component that calculates the diversity in the search results. Depending on the sensitivity level, the search results are either filtered out or are presented to the investigator.

Keywords: Privacy-enhancing technology, third party privacy protection

1. Introduction

Crimes involving computer equipment are investigated using procedures created specifically to surmount the difficulties associated with the digital nature of the crimes, and also to conform to the requirements for investigative procedures as may be required by law. In the discipline of digital forensics, there are typically two types of procedures, one geared for live investigations (investigations of crimes that are in

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 19–31, 2013.

progress), and the other targeted for *post mortem* investigations (investigations of crimes after they have been committed).

This paper is concerned with *post mortem* investigations, with a focus on multi-user environments. The investigative steps include the preservation, collection, examination and analysis of digital evidence [10, 15].

The collection phase involves the gathering of all digital data that may be relevant to the case. It includes the seizure of computing devices and the collection of data from storage media. The seized data normally pertains to the suspects and victims as well as third parties whose data and activities may have no bearing to the investigation. When an investigator examines collected data, he may stumble upon sensitive data relating to third parties. Even when filtering techniques are used, intrinsic or extrinsic connections between suspects/victims and third parties may result in third party data being scrutinized. This is an example of a third party privacy breach, where the act of investigating a crime inadvertently results in the loss of privacy for entities that are unrelated to the case.

This paper presents a privacy-enhancing framework that attempts to prevent – or at least minimize the risk of – third party privacy breaches. The approach draws on the basic privacy principle of information self-determination. In general, a privacy-enhancing technology (PET) protects access to personally identifiable information pertaining to the users of a computing system. During a digital forensic investigation, the reasonable expectation of privacy by entities that have no bearing to the case should not be discarded without due consideration [18]. Therefore, the framework is designed to protect third party privacy without preventing justifiable access to private information. The framework, which employs a filtering approach for weeding out data that is not relevant to an investigation, provides subjects in a multi-user environment with a reasonable expectation of privacy without reducing the effectiveness and speed of investigations.

2. Background

This section briefly discusses the main issues related to digital forensics, privacy and privacy-enhancing technologies (PETs).

2.1 Digital Forensics

Digital forensics is the branch of forensic science [10, 15] that is concerned with the "preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of evidence in a

digital context" [15]. Scientifically-sound and validated methods are required to be used in every phase of a digital forensic investigation.

During the collection phase, all digital information that may be relevant to the investigation is gathered [10, 17]. In practice, this often means creating bit-for-bit copies of the seized storage media. This is done to preserve the storage media content without any alteration. Typically, this is accomplished using a write-blocking device when copying the storage media.

Since all the data residing on the seized storage media is potential evidence, the investigator can be expected to access and examine all the data. In some jurisdictions, such as The Netherlands [18] and South Africa [13], data collected for investigative purposes is referred to as "police data." This data belongs to the police for investigative purposes. In both jurisdictions, the police are allowed to examine all police data without prejudice. Thus, the privacy rights of all the entities about whom there is data on the storage media are suspended during an investigation.

The analysis phase of a digital forensic investigation involves the systematic search of the collected information for data that is relevant to the case. This may involve searching through documents and other related files, searching the slack space and searching for data that may be hidden using anti-forensic techniques such as steganography [10, 17]. Since a large quantity of data has to be examined, a variety of digital forensic tools have been developed to support the analysis phase. Some of these tools (see, e.g., [8]) provide search filters that eliminate data that is considered to be noise (i.e., not relevant to an investigation). The filters typically use text strings (in documents or file metadata), file types and file timestamps [7].

Other techniques for filtering non-relevant data include text mining [2] and machine learning approaches such as self-organizing maps [7, 8]. Text mining [21] returns data that is important based on some relevance scale. Self-organizing maps [12] cluster data using various features to assist the investigator in forming opinions about the relevance of data. Noise may be filtered using the thematic clustering of search results [3].

The practice of filtering potential red herrings during the investigative process is well established. Filtering often yields false positives, i.e., data that may not be relevant to an investigation, which requires effort to classify and discard later in the investigation. False negatives are also a problem because they correspond to relevant data that is hidden from the investigator by the filtering tool.

2.2 Privacy

Efforts at protecting an individual's right to privacy have been undertaken for more than one hundred years [20]. Modern interpretations of the right to privacy are enshrined in the European Union Data Protection Directive [6] and the South African Protection of Personal Information Bill [13], that when enacted, will be on par with the European Union directive. Both these instruments deem privacy important enough to prohibit the cross-border flow of personal information to countries that do not have equivalent privacy laws. To reduce the trade impact of these laws, countries with strict privacy laws generally sign safe harbor treaties with non-compliant countries.

The principles governing the protection of private information were first listed by the Organization for Economic Cooperation and Development (OECD) [14]. The OECD principles, which are incorporated in the European Union and South African instruments, include collection limitation, data quality, purpose specification, use limitation, security safeguards, openness, individual participation and accountability. When implemented as technological safeguards for privacy protection, these principles are commonly translated to the right to information self-determination [9, 19] – the right of an entity to determine which entities have data about it, what data these other entities have, what these entities may do with the data, and to which entities they may give the data [19].

Legislation typically provides a reasonable expectation of privacy to individuals who entrust their data to enterprises and service providers who make positive statements about their intent to protect user privacy. The exemption of these laws during an investigation may take individuals by surprise – in particular, the fact that personal information that is not relevant to the case may be perused by an investigator during the course of the investigation.

2.3 Privacy-Enhancing Technologies

Privacy protection systems rely on PETs to protect the privacy of individuals. These systems focus on purpose specification and use limitation [9, 19]. While the systems address only two of the eight principles of privacy protection, the other six principles are normally taken care of as a matter of operating procedure. Privacy protection systems ensure that personal data is not disseminated to entities who are not supposed to get the data, and certainly not to entities who may desire to use the data for unauthorized purposes [19].

PETs are effective and, as such, are promising candidates for protecting personal privacy during digital forensic investigations. The challenge is to protect personal privacy without reducing the effectiveness and speed of investigations.

3. Preventing Third Party Privacy Breaches

As discussed above, filtering techniques can be used to exclude content that is deemed to be not relevant to an investigation. A privacy aware system used during an investigation would incorporate enhancements to basic filtering techniques. The important point is that the system would restrict the results to data that has limited potential to constitute a breach of third party privacy.

Croft and Olivier [5] articulated the notion of privacy aware digital forensics. In particular, they proposed the sequenced release of data to protect the privacy of individuals under investigation. Their proposal pivots on hypothesis testing before any additional data is released. In other words, the system only releases data if it knows that the entity querying the system already has some prior data that is relevant.

The approach presented in this paper allows the privacy of third parties to be taken into account. The system evaluates the results of a query and either releases the results to the investigator or reports that the query may result in a privacy breach. In the latter case, the investigator is requested to provide a more focused query.

Based on the operating principles of a PET listed above, it could be argued that the framework presented here is not a PET because no explicit purpose specification and use limitation are present. However, it is safe to assume that the purpose of the data access is the investigation, and that the data is stored and accessed for the purpose of the investigation.

The PET forces the investigator to present focused questions when searching for evidence. The search results are provided to the investigator only after it is determined that the query is focused. Clearly, this approach should not encourage data sequestration that may hinder an investigator (i.e., the system reports false negatives). In the case of false positives, the guiding principle is accountability. If personal data that is not related to the investigation is accessed, the PET should record the access, enabling the affected entity to know that a privacy breach did in fact occur.

A PET used in an investigative process should have the following characteristics:

- The PET should not encumber the investigator. The investigator should not have to perform a significant number of additional tasks during an investigation.
- The PET should reasonably sequester data that is determined as being not relevant to the investigation. It should specifically limit data related to parties who are not under investigation.
- The PET should provide the investigator with the freedom to access data that is deemed to be false negative.
- The PET should support accountability and openness. It should record accidental and intentional accesses of private data that bears no relevance to the investigation.
- The PET should not undermine an investigation by adding significant time delays to established investigative techniques such as filtering and text mining.

4. PET Design

Our PET design for supporting privacy aware digital forensic investigations incorporates four principal components (Figure 1). The first is a profiling component that creates a profile of the data and stores the results in a meta-database; the profile is created just in time, when a new document is queried and the profile is reused after it is created. The second component is a query front-end that enables an investigator to perform searches (e.g., partial string and file type searches). The third component is a response filter that determines if the query results could potentially lead to a privacy breach. The fourth component is an audit log that is used for auditing compliance with the privacy principles.

Note that the meta-database that stores profiling results and caches *ad hoc* profiles, and the audit log that stores search queries and PET configuration changes must be tamper-proof.

5. Screening Search Results

As stated above, an investigator may gain access to sensitive data concerning third parties during the course of an investigation. This problem is mitigated by incorporating a filter that forces the investigator to focus search queries on the case at hand. The solution engages the

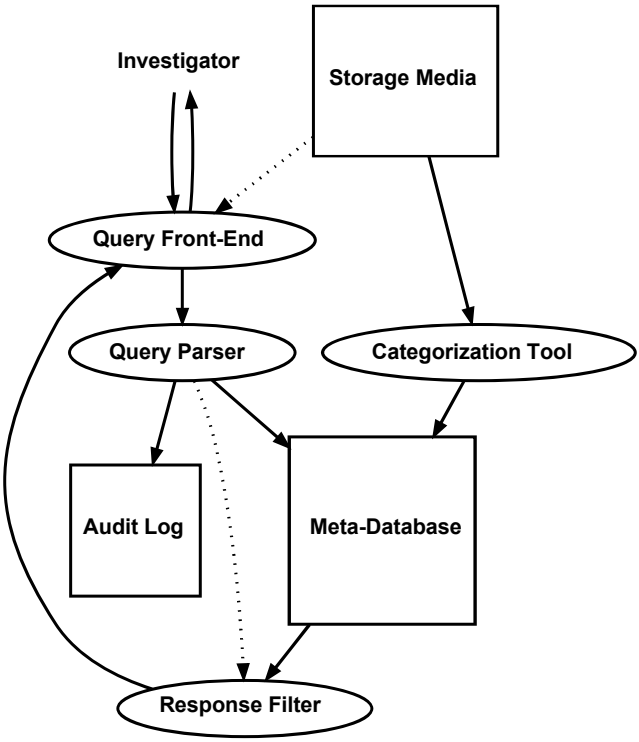


Figure 1. PET design for digital forensic investigations.

profiling component, a metric that assesses the diversity of search results and a sensitivity measure that determines the leniency to be provided to queries that return diverse search results.

5.1 Profiling

The profiling component uses n-grams [4], an established profiling technique that delivers good results. The n-gram technique relies on Zipf's law, which states that the n^{th} most common word in human text occurs with a frequency that is inversely proportional to n [4]. The implication is that two texts that are related should have similar frequency distributions. Using this principle, the text is tokenized and each token is deconstructed into n-grams where $2 \leq n \leq 5$ [4].

A frequency table is used for each n-gram in a text sample. After a document is scanned, a ranked list is created of the most frequently occurring n-grams. The number of entries in the rank table can vary; however, according to Cavnar and Trenkle [4], domain-specific terminol-

ogy only begins to appear around rank 300. The n-gram technique is fast and provides a usable profile for text.

After a profile is generated, it can be compared with another profile using a distance measure between the two profiles (or texts). Texts that are similar have a smaller distance measure than texts that are dissimilar. The next section describes how the distance is calculated.

5.2 Distance Calculation

Cavnar and Trenkle [4] use the rank-difference of n-grams as a distance measure:

$$\sum_{i}^{|P_k|} |r_{k(i)} - r_{j(i)}| \tag{1}$$

where P_k is the rank profile for text k and $r_{k(i)}$ is the rank of n-gram i for text k.

This distance measure takes content into account, but it ignores some important aspects. First, it is reasonable to assume that similar documents that have different owners are the result of two individuals working on the same project. Therefore, a difference in document owners should be considered when evaluating the similarity between documents. However, in an investigation, another user could be an accomplice of the suspect, so the existence of different owners should not skew the results too much.

Another aspect is the logical location of documents in a storage medium. This is important when two or more individuals work on a project, but only one of the individuals has been involved in malicious acts while working on the project. The goal here is to protect innocent individuals by carefully presenting the documents they own as separate and distant from the documents owned by the suspect.

The attributes used to determine the distance measure employ features used in anomaly detection by Fei, *et al.* [7]. In their approach, the documents that are clustered closely together using the features are worthy of further investigation.

The following features are considered for inclusion when determining the distances between the documents returned during a search for evidence: document text, document creation date and time, document owner and the absolute logical location of the document in the storage medium. Note that we do not propose a fixed set of features because the specific features used in distance calculations would depend on the type of investigation being conducted.

Thus, the distance between two documents in a search result corpus is calculated as:

$$D(j,k) = R(j,k) + |\delta_j - \delta_k| + P(j,k) + O(j,k) \tag{2}$$

where $R(j,k)$ is the rank-difference distance between j and k, δ_i is the creation date and time for i, $P(j,k)$ is the difference in the logical locations of the files, and $O(j,k)$ is the ownership distance. Furthermore, $P(j,k)$ is calculated as the number of directory changes needed to navigate from one location to the other.

The ownership difference $O(i,j)$ relies on a maximum value to create additional distance between the suspect and other users:

$$O(j,k) = \begin{cases} 0 & \text{if } j_o = k_o \\ \tau & \text{if } j_o \neq k_o \end{cases} \tag{3}$$

where τ is a maximum value assigned to $O(i,j)$.

After an investigator issues a query, the profiles of the documents are constructed (or retrieved). The distances between all the documents are then calculated. This information is used in the diversity calculation, which is discussed in the next section.

5.3 Diversity Classification

After a query is executed, the documents in the search results are clustered to determine the diversity of results using a hierarchical clustering scheme. Single-linkage clustering is used. However, the algorithm is only applied once, yielding an intuitive clustering result that provides a good indication of document similarity. This hierarchical clustering scheme also clusters documents very rapidly [11].

For a corpus S consisting of all the documents that are returned in a result:

$$y = argmin_{z \in S} D(x,z) \quad \forall x \in S, x \notin C_i \tag{4}$$

where $D(x,y)$ is a distance measure between x and y. Note that if $y \in C_i$ for some i, then $C_i \cup \{x\}$ else $C_j = \{x,y\}$ with $j = |C| + 1$.

Informally, for each document in the result, the cluster is found that most closely matches it, and the document is added to the cluster. If a document is found that most closely matches the document, but this document is not yet in any cluster, then a new cluster is created with the two documents.

After clustering is complete, the PET computes the diversity of the corpus using Shannon's entropy index [16]:

$$D_S = -\sum_{i}^{C} p_i log_2 p_i \tag{5}$$

where C is a cluster, i is a document in cluster C, and p_i is the probability of choosing document i from cluster C_i. The higher the entropy, the more diverse the corpus. Ideally, the search result would have returned one cluster of documents (i.e., all the documents are closely related).

This ideal search result, representing the lower bound on the diversity of the corpus, is called the "uniformity norm." It is calculated as:

$$N_S = -\sum_{i}^{S} p_i log_2 p_i \tag{6}$$

Ideally, a search result should be on par with ideal uniformity. In other words, the query is focused enough to return only documents that are closely related to each other. If this is not the case, the query should be rejected.

The proposed PET should support and not interfere with legitimate investigative efforts. To assess this, we introduce a sensitivity level indicator l as a threshold for determining if a search result is too diverse. If the diversity indicator $D_S > l$, then the query is defined as being "too wide" and the results are dropped. Note that l is range limited because $N_S \leq l$, which requires an ideal query result.

A query that returns results that are too diverse is referred to as a "wide query." On the other hand, a query that is close to the uniformity norm is referred to as a focused query. Note that all wide queries are reported and no results are returned.

A wide ranging implication with regard to threshold adjustment is that the threshold may be used without due regard for privacy and could be disregarded altogether. Specifically, if the threshold is set too low, then the attempt at preventing a third party privacy breach is disabled and all the results are returned. Cross-checks can serve to avoid this situation by requiring authorization from a higher authority before the threshold is adjusted. The audit logging component discussed in the next section is used to record threshold adjustments.

6. Audit Logging

The audit log records wide queries to ensure that only questions that are related to the investigation are asked and answered. Note that queries are not logged to have a "smoking gun" that the investigator issued unfocused queries, but rather to have documentary proof if the investigative process is questioned in the event of a privacy breach.

The audit log also records changes to the sensitivity level of the results filter, making it possible to trace privacy breaches at a later point in time. Because the queries are recorded and the data and meta-database

are kept intact, privacy breaches can be traced by examining the results obtained from queries logged in the system, similar to the auditing compliance work on Hippocratic databases done by Agrawal, *et al.* [1]. Additionally, the audit log provides a transparent recording of the investigative process.

7. Usability

Any digital forensic tool that casts doubt on the integrity of the evidence that it collects or processes is inappropriate for use in an investigation. The framework described in this paper is intended to be an extension of how an investigator searches for relevant data. It merely augments the analysis phase, attempting to prevent sensitive data about uninvolved third parties from being accessed. The framework does not change any data on the media being inspected, nor does it hinder the investigator or investigation. If the investigator uncovers data pertaining to a third party, the sensitivity level can be adjusted to reveal less data. Alternatively, the filter can be switched off completely, enabling the investigator to access all the data.

8. Conclusions

Preserving the privacy of uninvolved third parties is an important issue during a digital forensic investigation. The possibility of a privacy breach is especially high in multi-user environments where a suspect may have worked on group projects with other innocent users. The privacy protection framework described in this paper is highly effective during the *post mortem* data analysis phase of an investigation. One of its primary features is that an investigator is required to issue focused queries related to an investigation. If a query is deemed to be too wide, the framework prevents the investigator from accessing the search results. Should the investigator have concerns about missing relevant data, a sensitivity threshold can be adjusted to return results even if the query is considered to be unfocused. An audit log is maintained to ensure that queries and their results are saved for compliance purposes if a privacy breach were to occur. The framework operates on the same principle as tools that filter data that are not relevant to investigations. The important difference, however, is that the framework takes privacy concerns into account when filtering query results.

Our future research will evaluate the usability of the privacy protection framework in actual investigations. Also, it will examine the effectiveness of the framework when querying slack space data and file carving results.

References

[1] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau and R. Srikant, Auditing compliance with a Hippocratic database, *Proceedings of the Thirtieth International Conference on Very Large Databases*, pp. 516–527, 2004.

[2] N. Beebe and J. Clark, Dealing with terabyte data sets in digital investigations, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 3–16, 2005.

[3] N. Beebe and J. Clark, Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results, *Digital Investigation*, vol. 4(S), pp. S49–S54, 2007.

[4] W. Cavnar and J. Trenkle, N-gram-based text categorization, *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 161–175, 1994.

[5] N. Croft and M. Olivier, Sequenced release of privacy-accurate information in a forensic investigation, *Digital Investigation*, vol. 7(1-2), pp. 95–101, 2010.

[6] European Parliament and Council of the European Union, Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data, EU Data Protection Directive 95/46/EC, Brussels, Belgium, 1995.

[7] B. Fei, J. Eloff, M. Olivier, H. Tillwick and H. Venter, Using self-organizing maps for anomalous behavior detection in a computer forensic investigation, *Proceedings of the Fifth Annual South African Information Security Conference*, 2005.

[8] B. Fei, J. Eloff, H. Venter and M. Olivier, Exploring forensic data with self-organizing maps, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 113–123, 2005.

[9] S. Fischer-Hubner, *IT Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms*, Springer-Verlag, Berlin-Heidelberg, Germany, 2001.

[10] P. Gladyshev, Formalizing Event Reconstruction in Digital Investigations, Ph.D. Thesis, Department of Computer Science, University College Dublin, Dublin, Ireland, 2004.

[11] S. Johnson, Hierarchical clustering schemes, *Psychometrika*, vol. 32(3), pp. 241–254, 1967.

[12] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin-Heidelberg, Germany, 2001.

[13] Minister of Justice and Constitutional Development, Protection of Personal Information Bill, Pretoria, South Africa (`www.justice.gov.za/legislation/bills/B9-2009_ProtectionOfPersonalInformation.pdf`), 2009.

[14] Organization for Economic Cooperation and Development, OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, Technical Report, Paris, France, 1980.

[15] G. Palmer, A Road Map for Digital Forensic Research, DFRWS Technical Report DTR-T001-01 Final, Digital Forensic Research Workshop, Utica, New York (`www.dfrws.org/2001/dfrws-rm-final.pdf`), 2001.

[16] C. Shannon, A mathematical theory of communication, *Bell System Technical Journal*, vol. XXVII(3), pp. 379–423, 1948.

[17] Technical Working Group for the Examination of Digital Evidence, Forensic Examination of Digital Evidence: A Guide for Law Enforcement, Technical Report, National Institute of Justice, Washington, DC, 2004.

[18] R. van den Hoven van Genderen, Cyber Crime Investigation and the Protection of Personal Data and Privacy, Discussion Paper, Economic Crime Division, Council of Europe, Strasbourg, France, 2008.

[19] W. van Staden and M. Olivier, On compound purposes and compound reasons for enabling privacy, *Journal of Universal Computer Science*, vol. 17(3), pp 426–450, 2011.

[20] S. Warren and L. Brandeis, The right to privacy, *Harvard Law Review*, vol. IV(5), pp, 193–220, 1890.

[21] I. Witten, Text mining, in *Practical Handbook of Internet Computing*, M. Singh (Ed.), Chapman and Hall/CRC Press, Boca Raton, Florida, pp. 14-1–14.22, 2005.

Chapter 3

ON THE SCIENTIFIC MATURITY OF DIGITAL FORENSICS RESEARCH

Martin Olivier and Stefan Gruner

Abstract This paper applies a scientific maturity grade schema from the software engineering domain to research in the field of digital forensics. On the basis of this maturity schema and its grades, the paper classifies the current maturity of digital forensics research. The findings show that much more research conducted at higher levels of "scientificness" is necessary before the new field of digital forensics can be considered to be scientifically mature.

Keywords: Digital forensics, scientific maturity, software engineering

1. Introduction

The digital age has enabled and necessitated digital forensics as a means to maintain law and order in society. Forensic methods have occasionally failed those who were wrongfully convicted on the basis of low-quality evidence [14, 21]. In the past, the absence of strict scientific standards in some forensic practices has caused confusion about the reliability, validity, repeatability and accuracy of the outcomes, especially when the outcomes were presented in court, where the intended audience did not have the technical knowledge to judge the reliability of the presented evidence. Digital forensic scientists and practitioners have a duty to avoid repeating the mistakes of the past by scrutinizing the scientific maturity of their field, and by approaching and conveying evidence accordingly.

In a mature scientific field, the outcomes of processes can be trusted because they are constantly produced and reproduced in the course of "normal science." This trust is earned through the successful repeatability of processes, which render consistent, accurate, reliable and valid outcomes.

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 33–49, 2013.

Digital forensics is still a developing field, and the question arises: Is it possible to assess the current level of "scientificness" in the field of digital forensics? We use the informal term "scientificness" to acknowledge the fuzziness of the concept of scientific maturity.

This paper presents a scientific maturity assessment of digital forensics using the Shaw [45] software architecture science maturity scale. The transfer of this maturity scale, from software architecture to digital forensics, is justified by the similarity of the problems, including the shortage of scientificness (which was experienced in the domain of software architecture more than a decade ago) and the impact claims that both fields have in the physical world.

The remainder of this paper presents the software architecture scientificness scale [45]. The scale is used to assess the scientificness of the digital forensics research presented at the first and sixth *IFIP WG 11.9 International Conferences on Digital Forensics* held in 2005 and 2010, respectively. From the assessment, statements about the progress that has been made in the field of digital forensics in its brief history are discussed.

2. Scientificness Software Engineering

In 1998, Snelting [47] published a lament about the shortage of scientificness in the field of software engineering. Snelting reminded the software engineering community about Popper's criterion of falsifiability, with a "slant" against the post-modernist intellectual fashion of socio-constructivism, and demanded more efforts towards the empirical validation of ideas and conjectures. However, Snelting's appeal for more scientificness did not take into account the gradual historic development of emerging academic subjects from the pre-scientific stage through the proto-scientific stage to the fully scientific stage.

In a later paper focusing on the software engineering sub-specialty of software architecture, Shaw [45] included the spectrum of pre-scientific to fully scientific stages. With reference to earlier work by Redwine and Riddle, Shaw [45] identified six typical stages in the historic development of an emerging subject of research, especially in technical science domains. These six stages are characterized by their drive towards practical applications and external usefulness:

- Early prospecting.
- Concept formulation.
- Development and extension.
- Internal enhancement.

Table 1. Dimension 1: Research setting [45].

Dimension	Setting Type	Typical Questions
1.a	Feasibility	Is there an X? Is X possible at all?
1.b	Characterization	What is X like? What do we mean by X? What are the important characteristics of X? What are the varieties of X, and how are they related?
1.c	Capability	How can I accomplish X? Is there a smarter way of accomplishing X?
1.d	Generalization	Is X always true of Y? Given X, what is Y?
1.e	Valuation	Is X more desirable than Y? How do I decide?

- External enhancement.
- Popularization.

Most relevant to this study is Shaw's three-dimensional maturity classification scheme. The three dimensions are:

- Research setting.
- Research product (approaches and methods).
- Result validation technique.

Tables 1, 2 and 3 show the ascending maturity values in each of these three dimensions of assessments (including some interpretative modification and adaptation).

Table entries where the classification criteria differ from Shaw [45] use Bunge's terminology [3, 4]. The concept of tabulating levels of quality is also related to maturity assessment schemas in other domains, such as CMMI for general organizational capabilities and TMMI for maturity assessment in the domain of systematic software testing.

Shaw [45] concluded that researchers "must attend to making existing results more robust, more rigorously understood, and more ready to move into application." That is because, to date, "we don't recognize what our research strategies are and how they establish their results. Poor external understanding leads to lack of appreciation and respect. Poor internal understanding leads to poor execution, especially of validation, and poor appreciation of how much to expect from a project or result. There may also be secondary effects on the way we choose what problems to work on at all."

It is outside the scope of this paper to critique the maturity classification scheme. Instead, we transfer Shaw's maturity schema and method

Table 2. Dimension 2: Research product [45].

Dimension	Product Type	Typical Approach or Method
2.a	Qualitative or descriptive model	Organize and report interesting observations. Suggest and argue for generalizations from examples. Structure a problem area and formulate the right questions. Analyze a system or a project in an informal manner.
2.b	Technique	Invent new ways to do some tasks, including procedures and implementation techniques. Develop a procedure for choosing among alternatives.
2.c	System of knowledge or engineering	Embody results in a systematic context. Use system development as a source of insight and carrier of further results.
2.d	Empirical predictive model	Derive predictions from observed data.
2.e	Analytic model or theory	Develop structured quantitative and/or symbolic theories that permit formal analysis and deep explanations.

Table 3. Dimension 3: Validation technique [45].

Dimension	Technique Type	Style of Argument
3.a	Persuasion	We suggest... We believe...
3.b	Implementation	Here we made a prototype that can do... Here we see one example that has...
3.c	Informal evaluation	Comparison of several objects against each other. Rule-of-thumb comparison against check-lists. Exploratory measuring or counting without theoretical backup.
3.d	Formal analysis	Logical and/or mathematical proofs, including mathematical statistics.
3.e	Systematic experience	Theoretically motivated experiments. Reliable reproduction of previously hypothesized or predicted phenomena

of analysis to the domain of digital forensics, with the goal of revealing how research in the field of digital forensics is suffering from the same issue that software architecture research faced a decade ago. Consequently, we also argue that Shaw's conclusion and request for future work, as quoted above, can be transferred from the domain of software engineering to the domain of digital forensics today.

3. Scientific Maturity of Digital Forensics

The survey of the state of scientificness in digital forensics research is divided into two parts. First, a large statistical overview of 46 papers is given in terms of Shaw's model of scientific maturity [45]. Second, reviewer feedback for the early papers and some recent calls for developments in digital forensics are discussed in order to qualitatively deepen the findings from the statistical overview.

3.1 Classification of Conference Papers

This subsection analyzes the papers presented at two *IFIP WG 11.9 International Conferences on Digital Forensics* and subsequently published in the Springer book series, *Advances in Digital Forensics*. Shaw's model of scientific maturity [45] is used in the analysis.

For the purpose of discussion, it is sufficient to look at the first conference volume [39] from 2005 with 25 contributions, and one recent (sixth) conference volume, Volume VI [6] from 2010 with 21 contributions. The exercise of browsing through all the papers in the seven-volume series (at the time of conducting this research) would merely reinforce the argument because the results are similar for Volumes II–V and VII as well as for related journals and conference proceedings. Table 4 shows the raw data for this survey.

Table 4 and Figure 1 show that in both 2005 and 2010, the majority of papers do not reach Level **d** in any of the categories. However, a trend exists towards better evaluation efforts – with comparatively more work in Category 3.c – from 2005 to 2010. The large number of papers in Category 2.b in combination with Categories 1.a or 1.c indicates that many authors in the field of digital forensics are working in an engineering mode in which a useful solution is the goal, and not a theoretical explanation. Combinations of Categories 1.b and 2.c are rare in the table; this indicates a scientific research mode with an interest in knowledge for its own sake. Figure 1 shows the shrinking of the lowest value "—a—" in all three quality categories from 2005 to 2010. However, a high value "—d—" rarely appears in 2010.

The raw data in Table 1 is used to create the formal concept lattice visualization in Figure 2, which presents the mutual relationships existing between the papers in an intuitive manner. Figure 2 shows the diversity of attribute combinations (e.g., "1.a—2.b—3.b") extracted from Table 4, from all the papers in 2005 and 2010. The sixteen circles in the lattice in Figure 2 represent sixteen different attribute combinations. Small circles denote rare combinations while large circles denote frequently occurring combinations.

Table 4. Classification of *IFIP WG 11.9 Conference* papers.

Year 2005 (Early)		Year 2010 (Recent)	
Paper	**Classification**	**Paper**	**Classification**
[2]	1.a—2.a—3.a	[38]	1.a—2.a—3.a
[33]	1.b—2.a—3.c	[7]	1.b—2.a—3.a
[10]	1.a—2.b—3.b	[51]	1.a—2.a—3.a
[27]	1.a—2.a—3.a	[25]	1.c—2.b—3.c
[26]	1.b—2.a—3.c	[8]	1.c—2.a—3.c
[17]	1.c—2.b—3.b	[54]	1.c—2.b—3.b
[35]	1.b—2.b—3.c	[1]	1.c—2.c—3.c
[11]	1.c—2.b—3.**d**	[31]	1.a—2.a—3.c
[23]	1.a—2.a—3.a	[22]	1.b—2.b—3.c
[12]	1.c—2.b—3.b	[15]	1.c—2.b—3.c
[44]	1.c—2.b—3.b	[18]	1.c—2.b—3.c
[40]	1.a—2.a—3.b	[42]	1.a—2.b—3.c
[19]	1.b—2.b—3.b	[24]	1.a—2.b—3.c
[20]	1.b—2.b—3.b	[32]	1.a—2.b—3.c
[28]	1.a—2.a—3.a	[41]	1.c—2.b—3.c
[52]	1.b—2.b—3.c	[43]	1.c—2.b—3.b
[48]	1.a—2.b—3.b	[50]	1.**d**—2.c—3.**d**
[13]	1.c—2.b—3.b	[49]	1.c—2.b—3.c
[9]	1.c—2.b—3.a	[30]	1.a—2.b—3.c
[37]	1.a—2.a—3.a	[53]	1.c—2.b—3.b
[36]	1.c—2.b—3.c	[16]	1.b—2.a—3.a
[5]	1.c—2.b—3.c	-	-
[34]	1.c—2.b—3.b	-	-
[46]	1.a—2.a—3.a	-	-
[29]	1.a—2.a—3.a	-	-

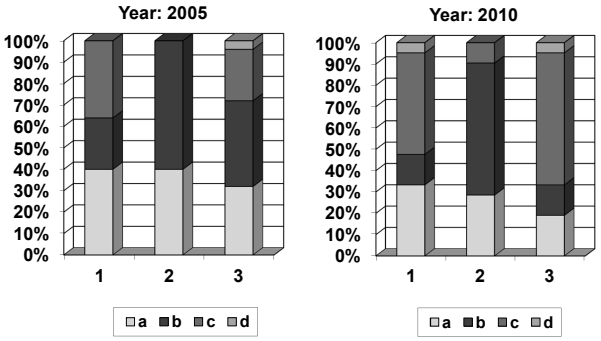


Figure 1. Shrinking of the lowest value in all three quality categories.

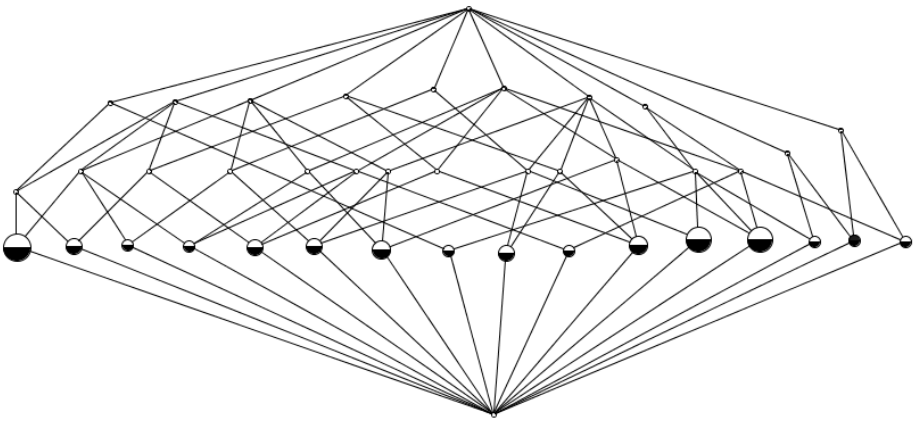


Figure 2. Method variety in the papers from 2005 and 2010.

Figure 2 shows a considerably large methodical variety in the rigor of the papers:

- The most frequent attribute combination is "1.a—2.a—3.a" (nine instance papers) represented by the large circle on the far left of the lattice in Figure 2. Papers with this attribute combination are suggestive, informal and new idea or proposal papers.
- The second most frequent combination is "1.c—2.b—3.b" (eight instance papers) represented by one of the two big circles on the right side of the lattice. These are typical engineering papers that focus on a useful skill and prove ability by pointing to a software implementation.
- The third most frequent combination is "1.c—2.b—3.c" (seven instance papers) represented by the other large circle on the right side of the lattice. These papers are also engineering-oriented, but they include additional efforts towards evaluating the properties of a software implementation, instead of merely presenting its existence as a proof-of-concept.

All together, these $9 + 8 + 7 = 24$ papers correspond to slightly more than 50% percent of the 46 papers analyzed in the two conferences.

3.2 Reviewer Perspectives

One of the authors of this paper served as the program co-chair of the second conference (in 2006). The reviews sent to the authors of accepted and rejected papers were still available. The feedback to the authors was

concatenated and scanned for comments that would confirm or refute the quantitative classification. Many reviews were positive, but only a few reviews suggested that the corresponding papers met the requirements of a mature discipline. Much of the positive feedback was based on the novelty of the idea in the paper, the way the paper extended the boundaries of digital forensics, or the fact that the paper covered a topic that was important in some way or another.

One reviewer, for example, stated that "despite its shortcomings, I believe the work is interesting for two reasons: it provides an opportunity for digital forensics people to take a look beyond the hard drive; and raises the important issue of" considering a specific facet of forensics during the development of new systems. It is a discipline finding its feet that commends worthwhile extensions and critiques those that seem unnecessary. Note that in quoting from a review, verbiage that may identify the paper is redacted.

As can be expected for a new discipline, most of the research feedback focuses on the boundaries of the field as well as initial explorations that show promise.

One positive comment was "This paper looks at a relevant problem and gives a simple solution for it." More critical are the comments that draw the boundaries closer than authors have hoped: "This is not a digital forensics topic. It is a computer security topic" or "The motivation for this work is not obvious." "The connection made to forensics, albeit not deniable, is quite tenuous" and "Despite the title, the paper does not address forensic issues" are two additional comments dealing with the boundaries of the discipline. Referring to Table 1, it is clear that these comments are at the low end of the scale.

It should be noted that a good number of papers explore questions of the form "What is X?" and more often "Is X possible?" (Table 1). Each of these papers typically elicited a statement from a reviewer that the paper shows that X has potential. As such, the papers met reviewer expectations, although the reviewers often wanted tools that could shift the boundaries. In other words, proposing a system to achieve X was acceptable. However, with X as the research product, some additional functionality was expected, which is indicative of an increase in the research setting scale.

3.3 Research Product

The research product dimension included the most critical feedback. The community mainly comprises individuals who often have one foot in academia and the other in practice. The practitioners want tools that

provide a competitive advantage. However, even from a pure scientific standpoint, research products (new tools, techniques and theories) are likely to push the research setting scale to higher levels.

Examples of comments dealing with the research product dimension are:

- "The primary flaw with the paper is that the work is not presented in enough detail for the underlying technology to be used or replicated."
- "This is not really a research paper. It is more of a hands-on lab manual."
- "Instead of discussing how this tool could benefit the area of digital forensics, the author focused on how the tool is built and functions."
- "It would have been nice to see an analysis of an implementation, what were the end requirements of the investigator for initializing and using the system on a compromised network. As well as a study on the amount and type of logging that is necessary."
- "The paper is definitely one I would like to see developed, but as the document is I found it hard to find significant value."
- "This paper presents a sketch of an architecture for recording and retrieving TCP/IP network data in a ... system."
- "No new material is contributed and some vital current methods/techniques for establishing location during a network event are missing."
- "There is no mention of how these things will be done other than stating that they are future work, which really should have been done."
- "It is more like a position paper."
- "In terms of pure computer science, this is yet another file format and I find the basic design reasonable."

The (explicit or implied) critique above is of the form that this is "yet another" solution to a known problem. The fact that "yet another" tool or technique has been developed is not necessarily bad. In an emerging field, it may be fatal to prematurely suppress alternative new ideas. Solutions that venture beyond the beaten track may be the ones that eventually have broad impact.

However, a number of solutions were critiqued in that the mere fact that they were novel was no longer sufficient. Many of the comments called for a greater emphasis in the validation technique dimension rather than a deepening of the research product dimension:

- "Also, the paper should have included experimental results to make it more convincing and solid."
- "My question is how do you arrive at 30% and 15%?"
- "At a minimum, the authors should convince the reader that there are many things mentioned in the paper that are technology invariant."
- "We should have a good start on [some specific forensic issue] but are not working on it from the scratch."

3.4 Validation Technique

The reviewer feedback reflects a number of issues regarding the validation dimension described in Table 3.

One recurring theme was doubt that a solution was correct. One reviewer stated "the analytical results in the paper heavily rely on [some] assumption. Without this assumption, the analytical results would not be valid. But in most real-world scenarios, this assumption is invalid." A related remark about a different paper questions the data used, rather than an assumption: "I would like to be convinced that this is not a toy or contrived problem. The authors could do this by validating their algorithm on actual data, rather than on generated data."

Some other attempts at validation were not met with the same skepticism, but pointed out that the validation was incomplete in some respect. For example: "One of the key aspects of this paper is the new ... protocol, which even the author says has not been proven to work as advertised." In another instance, a reviewer laments "Currently the paper does not provide any evidence that such a relationship exists." In one case, the absence of validation is noted: "One thing lacking is a validation of the model."

As a final illustration, consider the following remarks about missing details, which primarily affect repeatability:

- "My main problem with this paper is that some very important details are missing. The authors talk about quantifying the confidence of a forensic examination, but give no information whatsoever how this quantification is done."

- "The main problem with the paper is its relative vagueness."
- "What is the standard deviation of the frequencies and other results?"

4. Stability and Fluctuation of the Community

The maturation of an emerging field of research is also related to the stability and fluctuation of the community of researchers who are active in a field. The arguments encouraging these aspects are:

- Serious researchers, who have confidence in the worthiness of their own work, do not tend to abandon their field of work prematurely.
- New researchers tend to flow into an emerging research field as others begin to recognize the relevance of the field.
- Some fluctuation is to be expected by the natural retirement of the pioneers and "founding fathers."
- Short-term fluctuation is to be expected due to the co-authorship of student-researchers, who often depart the research scene after having obtained their postgraduate degrees.
- The long-term existence of a small set of "gurus" and the high fluctuation rate of student co-authors are likely symptoms of esoteric stagnation and lack of external popularization of the field.

To this end, we have computed a formal concept lattice that represents the "community" during the years 2005 and 2010, based on the authorship of all the papers from the two conferences. Figure 3 shows a lattice graph of the overall author community, with the 2005 community on the left-hand side and the 2010 on the right-hand side. Research clusters are marked using circles, and the middle-ground represents a continuity of researchers who co-authored papers in 2005 and 2010.

The lattice graph shows rapid fluctuation in the community of co-authors during the relatively short period of time between 2005 and 2010, with only a small number of co-authors present in both 2005 and 2010. Some clusters of researchers are recognizable, but on the other hand, there are also a considerable number of contributions from outside the clusters. In other words, the research community depicted in Figure 3 does not suffer from unhealthy inbreeding. On the other hand, the rather small area of personal continuity during a short time span of only five years might be a cause for concern.

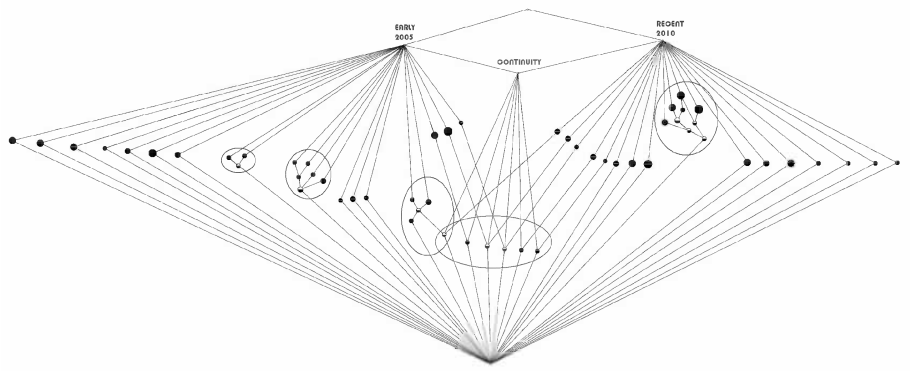


Figure 3. Lattice showing the early (2005) and recent (2010) author communities.

5. Conclusions

This paper has attempted to evaluate the scientific maturity of digital forensics. The statistical review and the qualitative remarks demonstrate that the lack of scientificness that characterized software engineering a decade ago is currently present in digital forensics research. Like Shaw [45] did in the case of software engineering, we emphasize that digital forensics must become more scientific and we urge our colleagues to redouble their efforts to increase the level of scientificness. We also believe that Shaw's model provides a strategy for incrementally improving the scientificness of digital forensics research to the point where the discipline can be considered to be scientifically mature.

Acknowledgements

The authors thank Candice le Sueur for her assistance with writing the introductory remarks. The authors also thank the anonymous reviewers and workshop participants for their constructive comments.

References

[1] S. Al-Kuwari and S. Wolthusen, Forensic tracking and mobility prediction in vehicular networks, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 91–105, 2010.

[2] N. Beebe and J. Clark, Dealing with terabyte data sets in digital investigations, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 3–16, 2005.

[3] M. Bunge, *Philosophy of Science (Volume One): From Problem to Theory*, Transaction Publishers, New Brunswick, New Jersey, 1998.

[4] M. Bunge, *Philosophy of Science (Volume Two): From Explanation to Justification*, Transaction Publishers, New Brunswick, New Jersey, 1998.

[5] Y. Chen, V. Roussev, G. Richard and Y. Gao, Content-based image retrieval for digital forensics, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 271–282, 2005.

[6] K. Chow and S. Shenoi, *Advances in Digital Forensics VI*, Springer, Heidelberg, Germany, 2010.

[7] F. Cohen, Toward a science of digital forensic evidence examination, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 17–35, 2010.

[8] S. Conrad, G. Dorn and P. Craiger, Forensic analysis of a Playstation-3 console, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 65–76, 2010.

[9] P. Craiger, Recovering digital evidence from Linux systems, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 233–244, 2005.

[10] M. Davis, G. Manes and S. Shenoi, A network-based architecture for storing digital evidence, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 33–42, 2005.

[11] T. Duval, B. Jouga and L. Roger, The Mitnick case: How Bayes could have helped, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 91–104, 2005.

[12] B. Fei, J. Eloff, H. Venter and M. Olivier, Exploring forensic data with self-organizing maps, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 113–123, 2005.

[13] P. Gershteyn, M. Davis, G. Manes and S. Shenoi, Extracting concealed data from BIOS chips, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 217–230, 2005

[14] P. Giannelli, Wrongful Convictions and Forensic Science: The Need to Regulate Crime Labs, Working Paper 08-02, School of Law, Case Western Reserve University, Cleveland, Ohio, 2008.

[15] M. Gunestas, M. Mehmet and D. Wijsekera, Detecting Ponzi and pyramid business schemes in choreographed web services, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 133–150, 2010.

[16] Y. Guo and J. Slay, Data recovery function testing for digital forensic tools, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 297–311, 2010.

[17] M. Hoeschele and M. Rogers, Detecting social engineering, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 67–77, 2005.

[18] R. Ieong, P. Lai, K. Chow, M. Kwan and F. Law, Identifying first seeders in Foxy peer-to-peer networks, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 151–168, 2010.

[19] P. Kahai, M. Srinivasan, K. Namuduri and R. Pendse, Forensic profiling system, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 153–164, 2005.

[20] E. Kim, D. Massey and I. Ray, Global Internet routing forensics, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 165–176, 2005.

[21] R. Koppl and M. Ferraro, Digital devices and miscarriages of justice, *Daily Caller* (`dailycaller.com/2012/06/15/digital-devices-and-miscarriages-of-justice`), June 15, 2012.

[22] M. Kwan, R. Overill, K. Chow, J. Silomon, H. Tse, F. Law and P. Lai, Evaluation of evidence in Internet auction fraud investigations, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 121–132, 2010.

[23] R. Laubscher, D. Rabe, M. Olivier, J. Eloff and H. Venter, Applying forensic principles to computer-based assessment, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 105–112, 2005.

[24] F. Law, P. Chan, S. Yiu, B. Tang, P. Lai, K. Chow, R. Ieong, M. Kwan, W. Hon and L. Hui, Identifying volatile data from multiple memory dumps in live forensics, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 185–194, 2010.

[25] F. Li, H. Chan, K. Chow and P. Lai, An analysis of the Green Dam Youth Escort Software, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 49–62, 2010.

[26] M. Losavio, Non-technical manipulation of digital data, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 51–63, 2005.

[27] M. Meyers and M. Rogers, Digital forensics: Meeting the challenges of scientific evidence, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 43–50, 2005.

[28] T. Moore, A. Meehan, G. Manes and S. Shenoi, Using signaling information in telecom network forensics, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 177–188, 2005.

[29] Y. Motora and B. Irwin, In-kernel cryptographic executable verification, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 303–313, 2005.

[30] Y. Nakayama, S. Shibaguchi and K. Okada, A visualization system for analyzing information leakage, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 269–282, 2010.

[31] S. Ngobeni, H. Venter and I. Burke, A forensic readiness model for wireless networks, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 107–117, 2010.

[32] J. Okolica and G. Peterson, A compiled memory analysis tool, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 195–204, 2010.

[33] M. Olivier, Forensics and privacy-enhancing technologies, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 17–31, 2005.

[34] L. Peng, T. Wingfield, D. Wijsekera, E. Frye, R. Jackson and J. Michael, Making decisions about legal responses to cyber attacks, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 283–294, 2005.

[35] A. Persaud and Y. Guan, A framework for email investigations, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 79–90, 2005.

[36] G. Peterson, Forensic analysis of digital image tampering, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 259–270, 2005.

[37] S. Piper, M. Davis, G. Manes and S. Shenoi, Detecting hidden data in ext2/ext3 file systems, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 245–256, 2005.

[38] M. Pollitt, A history of digital forensics, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 3–15, 2010.

[39] M. Pollitt and S. Shenoi, *Advances in Digital Forensics*, Springer, Boston, Massachusetts, 2005.

[40] S. Redding, Using peer-to-peer technology for network forensics, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 141–152, 2005.

[41] V. Roussev, Data fingerprinting with similarity digests, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 207–226, 2010.

[42] A. Savoldi, P. Gubian and I. Echizen, Uncertainty in live forensics, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 171–184, 2010.

[43] B. Schatz and M. Cohen, Redefining evidence containers for provenance and accurate data representation, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 227–242, 2010.

[44] K. Shanmugasundaram, H. Bronnimann and N. Memon, Integrating digital forensics in network infrastructures, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 127–140, 2005.

[45] M. Shaw, The coming-of-age of software architecture research, *Proceedings of the Twenty-Third International Conference on Software Engineering*, pp. 656–664, 2001.

[46] J. Slay and K. Jorgensen, Applying filter clusters to reduce search state space, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 295–301, 2005.

[47] G. Snelting, Paul Feyerabend und die Softwaretechnologie, *Informatik Spektrum*, vol. 21(5), pp. 273–276, 1998.

[48] C. Swenson, G. Manes and S. Shenoi, Imaging and analysis of GSM SIM cards, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 205–216, 2005.

[49] K. Tadano, M. Kawato, R. Furukawa, F. Machida and Y. Maeno, Digital watermarking of virtual machine images, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 257–268, 2010.

[50] V. Thing, Virtual expansion of rainbow tables, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 243–256, 2010.

[51] K. Wang, Using a local search warrant to acquire evidence stored overseas via the Internet, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 37–48, 2010.

[52] S. Willassen, Forensic analysis of mobile phone internal memory, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 191–204, 2005.

[53] Y. Yang, K. Chow, L. Hui, C. Wang, L. Chen, Z. Chen and J. Chen, Forensic analysis of popular Chinese Internet applications, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 285–295, 2010.

[54] Y. Zhu, J. James and P. Gladyshev, A consistency study of the Windows registry, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 77–90, 2010.

II

FORENSIC MODELS

Chapter 4

COGNITIVE APPROACHES FOR DIGITAL FORENSIC READINESS PLANNING

Antonio Pooe and Les Labuschagne

Abstract This paper focuses on the use of cognitive approaches for digital forensic readiness planning. Research has revealed that a well-thought-out and legally contextualized digital forensic readiness strategy can provide organizations with an increased ability to respond to security incidents while maintaining the integrity of the evidence gathered and keeping investigative costs low. This paper contributes to the body of knowledge in digital forensics related to the design and implementation of digital forensic readiness plans aimed at maximizing the use of digital evidence in organizations. The study uses interviews as part of a mixed-methods approach. In particular, it employs a mix of informal conversational and standardized open-ended interview styles conducted with industry experts over a variety of communication media.

Keywords: Digital forensic readiness, digital evidence, cognitive approaches

1. Introduction

From the perspective of law enforcement agencies, the forensic process begins when a crime has been committed or when a crime has been discovered and reported [6]. Forensic readiness enables organizations to preempt the occurrence of crimes by gathering evidence in advance and, in doing so, derive benefits in instances where prosecution becomes an issue and limit their risks [5].

The organizational requirement to gather and use digital evidence has been recognized in a number of studies (see, e.g., [2, 5]). These studies stress the importance of a structure to maintain the integrity of forensic evidence. In particular, Yasinsac and Manzano [7] note that organizational policies play a critical role in providing the needed structure. Yasinsac and Manzano also propose six categories of policies to facili-

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 53–66, 2013.

tate digital forensic investigations. The categories are designed to help organizations deter digital crime and position themselves to respond to attacks by improving their ability to conduct investigations. The six categories of policies that facilitate digital forensic investigations are:

1. **Retaining Information:** Policies that relate to the storage of information by an organization.
2. **Planning the Response:** Policies that guide an organization's plans for responding to incidents and situations.
3. **Training:** Policies that address the training of staff members and others affiliated with an organization.
4. **Accelerating the Investigation:** Policies that address the operational aspects of investigations.
5. **Preventing Anonymous Activities:** Policies that address an organization's proactive efforts against fraud.
6. **Protecting the Evidence:** Policies that address the handling and protection of evidence and other vital data.

From the above discussion, the concept of digital forensic readiness has two main objectives: (i) maximizing the ability to collect credible digital evidence (Categories 1, 2, 5 and 6 above); and (ii) minimizing the cost of digital forensics during incident response (Categories 3 and 4). While this reinforces the importance of cohesive policies in organizations, the problem with the categorization is that it suggests that organizations must have all six policies in place, which may result in possible duplication and/or conflicting policy statements. Furthermore, it may lead to confusion in identifying the authority/governing policy for facilitating digital investigations. While the policies are important, they alone do not guarantee a holistic digital forensic readiness plan.

Because of the potential policy conflicts, our study used mixed methods interviews [4] as a means to develop a holistic digital forensic readiness plan. The interviews were employed as an exploratory research tool to gather information from subject matter experts and to capture real-world experiences with the goal of identifying key components for consideration.

2. Research Design

Mixed method research is a design with philosophical assumptions and various methods of inquiry [4]. The philosophical assumptions guide the

direction of the collection and analysis of data while the combination of qualitative and quantitative methods of inquiry in a single or series of studies offers a better understanding of research problems than each approach on its own [12].

The intent of the two-phase exploratory design is that the results of the first method (qualitative) can help develop or inform the second method (quantitative) [4]. This is based on the premise that an exploration may be needed for one or more reasons, which takes into account the possibility that measures or instruments are not available, variables are unknown and no guiding framework or theory exists.

This design is used because it enables the exploration of a phenomenon in detail and the development and testing of the resulting conceptual model [4, 12]. The use of the design in this study helps validate qualitative data with quantitative results.

Interviews were used as the data collection method. In mixed methods research, open-ended qualitative interviews (INT-QUAL) are featured more frequently than closed-ended quantitative interviews (INT-QUAN). Qualitative interviews are usually non-directive and general ("tell me about your school"). On the other hand, quantitative interviews are structured and closed-ended ("which of the following describes the food in your school cafeteria – very good, good, bad, very bad") [12].

2.1 Types of Interviews

Patton [8] defined four types of open-ended interviews, ranging from the least structured (informal conversational interviews) to the more structured (general interview-guided approaches) to the most structured (standardized open-ended interviews). He also described closed fixed-response interviews but does not advocate their use. The four types of open-ended interviews are:

- **Type 1: Informal Conversational Interviews:** Questions emerge from the immediate context and are asked in the natural course of the interview. The question topics and wording are not predetermined.

- **Type 2: General Interview Guide Approaches:** Topics and issues are specified in advance in outline form. The interviewer decides the sequence and working of questions in the course of the interview.

- **Type 3: Standardized Open-Ended Interviews:** The exact wording and sequence of questions are determined in advance. All

the interviewees are asked the same basic questions in the same order. Questions are worded in a completely open-ended format.

- **Type 4: Closed Fixed-Response Interviews:** Questions and response categories are determined in advance. The responses are fixed. The respondent chooses from among the fixed responses.

For purposes of this study, a mixture of Type 1 and Type 3 open-ended interviews was used. Teddie and Tashakkori [12] state that researchers who select the INT-QUAL strategy may use any of the open-ended interview approaches and potentially combine the interview types. They suggest the following sequence of interview techniques:

- Start with the unstructured informal conversational interview approach to build rapport and elicit spontaneous responses.
- Move to the interview guide approach, which provides a more comprehensive outline of topics, but still maintains a conversational tone.
- Finish with the highly structured, standardized open-ended interview approach, which greatly increases response comparability.

Our study began with an unstructured informal conversational interview approach, followed by a highly structured, standardized open-ended interview approach. The questions were formulated based on a literature survey conducted in 2011 [9].

2.2 Interviews

This section discusses the criteria used to select the interviewees, the communication channels used to conduct the interviews and the ethical considerations related to the interview process.

The interviewees were selected based on three criteria:

- Individuals from the private sector and law enforcement were selected in order to emphasize the multi-disciplinary aspects of the domain and to capture a broad range of views from subject matter experts involved in different aspects of the digital forensic process.
- Individuals with experience in digital law and/or digital forensics were selected to ensure that the input gathered was not biased and addressed the technical and legal dimensions of digital forensics.
- Individuals who had been practicing digital forensics in South Africa for a period of no less than three years were selected. Since

Table 1. Interviewee profiles.

Interviewee	Experience > 3 Years	Law Enforcement	Private Sector	Management Position
INT1	Yes	Yes	No	Yes
INT2	Yes	Yes	Yes	Yes
INT3	Yes	No	Yes	Yes
INT4	Yes	Yes	Yes	Yes
INT5	Yes	No	Yes	Yes
INT6	Yes	Yes	No	Yes
INT7	Yes	Yes	Yes	Yes

the context of the study was South Africa, it was important to identify subject matter experts with experience in the geographical context.

These criteria ensured that the interviewees would provide a mixture of opinions based on their experiences in their different working environments.

Studies have shown that, while open-ended interviews are typically conducted in a face-to-face manner, they may also be conducted by telephone or over the Internet [11, 12].

A total of seven interviews were conducted using three channels: four interviews were conducted face-to-face, one was conducted over the phone and two over the Internet. Due to the volume of data collected during the interviews, the ATLAS.ti tool [1] was used to process and analyze the data.

3. Interview Results

This section describes the results of the seven interviews. All the interviewees met the selection criteria. Table 1 summarizes the interviewee profiles.

We now provide a summary of some of the questions, the responses received and the interviewees that were in agreement. Based on the responses, cognitive approaches to digital forensic readiness planning were used to develop a conceptual model.

- *Question 1: Should South African organizations be concerned about digital crimes?*

  The responses to this question revealed the following opinions:

  - Digital crimes are on the increase (All).

  - The intangible nature of data in an electronic format causes people to lower their defenses (INT1, INT4, INT7).
  - Modern criminals are technologically literate and have access to good legal representation (INT1, INT2, INT3).
  - Immaturity of the digital forensics profession allows criminals to go free (INT5).
  - Following correct investigative processes to preserve evidence is important (INT6).

- *Question 2: Which three types of digital crimes do you find to be the most prevalent?*

  The following crimes were found to be prevalent:

  - Financial crimes (All).
  - Child pornography (INT4, INT5, INT6, INT7).
  - 419 scams (INT4, INT6).
  - Malware-related crimes (INT1, INT2).
  - Intellectual property theft (INT1, INT5).
  - Hacking and illegal access (INT2, INT3).
  - Internet misuse (INT5).

- *Question 3: Which sector do you find to be the most targeted?*

  The responses indicated the following sectors:

  - Banking/financial sector (All).
  - Large corporations (INT2, INT4, INT7).
  - Individuals (INT2, INT3).
  - Mining (INT4).
  - Businesses (INT5).

- *Question 4: Have you noted any challenges regarding the prosecution of digital crimes?*

  The following challenges were noted:

  - Knowledge of digital forensic principles is lacking among the stakeholders (All).
  - Lack of understanding of legal requirements (INT1, INT3, INT4, INT5, INT6).
  - Lack of resources (INT2, INT7).

- *Question 5: Do you or your organization have a digital forensic model that has been adopted?*

  All the respondents indicated that they use their own entity-specific model, which may differ from those used by other organizations.

- *Question 6: Does electronic evidence provide sufficient assurance of non-manipulation?*

  Provided that the correct processes were followed, all the respondents were of the opinion that electronic evidence can be relied upon.

- *Question 7: Is there a standard process for electronic evidence gathering?*

  All the respondents indicated that, while processes adopted in their individual organizations were similar, no process standards specific to South Africa exist.

- *Question 8: Does the law adequately position the acceptable use of and/or extent to which electronic evidence can be used in civil or criminal proceedings?*

  All the respondents referred to the Electronic Communications and Transactions (ECT) Act of 2002 as legislation that makes it possible to present electronic evidence in a South African court of law. However, the following contradictions were noted:

  - Existing laws support the ECT Act (INT4, INT5, INT7).
  - Discrepancies exist between the ECT Act and existing laws (INT2).

- *Question 9: Does the law cater to the complexities of modern IT devices?*

  All the respondents indicated that the law lagged behind technology. While the ECT Act was found to be strong legislation, the respondents indicated that it needed periodic review.

- *Question 10: What are the factors that contribute to electronic evidence being rendered inadmissible?*

  All the respondents pointed to digital forensic processes and procedures as a good foundation for ensuring the admissibility of evidence.

- *Question 11: Have you noted any challenges that prevent digital crime investigators from correctly applying a digital forensic model or framework?*

All the respondents indicated that a single point of reference was needed. Other points noted were:

- South Africa needs a specific model (INT5, INT7).
- The model must enable and support legal processes (INT7).
- The model must be flexible (INT5).

- *Question 12: Do you think digital forensic investigators are sufficiently trained to do their work?*

  All the respondents identified a need for more training of local digital forensic investigators.

- *Question 13: Have you noted any challenges that prevent prosecutors from successfully prosecuting digital crimes?*

  The responses identified the following challenges:

  - Lack of interest in digital crimes (INT1, INT3, INT4, INT5, INT6, INT7).
  - High case loads (INT1, INT2, INT4, INT5, INT7).
  - Lack of digital forensic training and awareness (INT2, INT5, INT7).
  - Lack of cooperation (INT5).

- *Question 14: Do you think state prosecutors are sufficiently trained to do their work?*

  All the respondents opined that a training need exists for state prosecutors.

- *Question 15: What do you think should be done to increase the prosecution rate of digital crimes in South Africa?*

  The responses included:

  - Special digital forensic courts (INT1, INT3, INT4, INT5, INT7).
  - More digital forensic education, training and awareness (INT1, INT2, INT6, INT7).
  - More research focused on digital forensics (INT1, INT6).
  - Compulsory reporting requirements (INT1, INT5).
  - A new law of evidence for electronic crimes (INT2).
  - Define processes and a model (INT4).

4. Data Analysis and Interpretation

This section discusses the application of the data analysis method and presents the results of the analysis.

The data analysis was conducted by transcribing each interview and reading each transcript repeatedly to identify the codes for each question answered by the interviewees. Techniques from immersion/crystallization and constant comparison (grounded theory) were applied to assist in developing the initial and final codes [3]. The iterative reading process made it possible for focus/immersion to be applied to each question and for the emergence/crystallization of themes to take place. Differences in findings (codes and themes) were also resolved using the iterative process. The process resulted in three environments: corporate, industry and legislative.

4.1 Corporate Environment

The correlation of the interview results yielded the following key findings and analyses:

- **Organizational Culture:** A large proportion of organizations were found to have a habit/culture of ignoring/overlooking small crimes. Additionally, organizations were found to invest the least amount of resources to address the risk of digital crimes. Also, a lack of awareness about digital crime prevention and detection was found to exist.

  These findings suggest that, by creating a culture of no tolerance to crime and taking action on reported crimes, an organization can significantly reduce its risk exposure to digital crimes.

- **Policies:** Interviewees noted that a general lack of governance, policies and procedures relating to fraud risk management existed in many of the organizations with which they had been in contact.

  These findings support existing literature (as discussed earlier in this study) on the importance of policies as they relate to achieving digital forensic readiness.

- **Communication Channels:** The reporting of digital crimes was found to be low. The causes mentioned included a culture of "sweeping things under the carpet" along with ignorance, and the lack of education and law enforcement effectiveness.

  These findings suggest that encouraging and supporting open dialog, coupled with crime reporting mechanisms can positively impact organizational culture.

- **Emerging Risks:** The following areas of risk were identified:
  - The intangible aspect of technology causes people to lower their defenses. This is evident in the various types of white-collar crimes committed using technology.
  - Modern criminals are technologically literate and have access to financial and other resources, including good legal representation.
  - Criminals are quick to exploit innovations in mobile technology.
  - The digital forensic industry is not sufficiently mature, enabling criminals to take advantage of ambiguities in global legislative structures.
  - Digital forensic investigators often do not follow due process, which contributes to the low prosecution rate of digital crimes.

  The findings suggest the importance of being cognizant about emerging risks because they affect the nature of controls and mitigation strategies that organizations employ.

- **Crime Trends:** Respondents were of the opinion that digital crimes are on the increase and will affect all current and future users of technology. The high prevalence of crime is attributed to legislative gaps.

These findings suggest that increased awareness of crime trends can aid organizations in focusing their attention and resources on high risk areas.

4.2 Industry Environment

The correlation of the interview results yielded the following key findings and analyses:

- **Standards:** The absence of standards for digital forensics was identified as an inhibiting factor to the prosecution of digital crimes. An open culture of information sharing was noted as necessary to promote the maturity of the digital forensic profession. Specific to a digital forensic model, this should not be legislation but, instead, recommended guidelines that enable and support the legal process while being sufficiently flexible to accommodate advances and changes in legislation and technology.

These findings suggest that establishing governance structures is an important step to building quality digital forensic case law and professionalizing the digital forensic industry.

- **Methodology:** While digital forensic investigation methodologies exist, there is a need for a single point of reference. This extends to the need for consistency in country-specific standards, processes and methodology. Awareness of research-based methodologies and their alignment to legislation was also found to be necessary.

  These findings suggest that standardization can encourage consistency, conformity, compliance and increase competitiveness in the digital forensic profession.

- **Education:** A lack of cohesion between academics and practitioners was found to exist. While interviewees noted that no single qualification is a prerequisite to qualify as a digital forensic practitioner, specialized training and a balance of education and experience were found to be necessary. A digital forensic model was also found to be essential to guide training efforts.

  These findings suggest that the ideal qualification requirements for digital forensic practitioners are formal education coupled with advanced training in a field of specialization.

- **Training:** Training of all stakeholders was noted as essential. In particular, emphasis was placed on legal requirements as superseding technical processes. Some practitioners lack an understanding of the legal requirements and/or incorrectly apply technical knowledge.

  These findings suggest the need to expose digital forensic investigators to a balanced training curriculum that covers all related disciplines.

- **Development:** There was a lack of continued professional development for individuals in the investigation value chain, including prosecutors, judges and digital forensics practitioners across all sectors.

  These findings suggest the need for organizations to employ qualified digital forensic investigators and provide continuous education and career development in order to ensure that investigators are equipped with the skills necessary to handle modern digital crimes.

4.3 Legislative Environment

The correlation of the interview results yielded the following key findings and analyses:

- **Legislative Culture:** A lack of consideration of digital crimes exists among some judges and prosecutors; this was attributed to high caseloads and limited resources. The creation of special interest groups and special courts could introduce positive changes.

  These findings suggest the need for a culture of cooperation between organizations, law enforcement and prosecuting authorities.

- **eLaw Development:** Specific to South African legislation on electronic evidence [10], the findings indicate that the key strengths of the law are robust legislation and good baselines. On the other hand, the findings also suggest that the law imposes low penalties, there is limited awareness and understanding of the law, and considerable complexity in its use of technical terminology. Lastly, the South African legislation on electronic evidence was perceived as being adequate to cover technological advances over the next ten to twenty years.

  These findings suggest the need for continuous review and development of legislation relating to digital crimes in order to close the gap (of relevance) between technology and the law.

- **Awareness:** The findings suggest that the judiciary is presented with challenges that result in a reluctance to prioritize digital evidence. High caseloads and a lack of awareness of digital forensic processes and methodologies exist. Interviewees pointed to a need for a change in culture. Training and awareness were suggested as effective drivers to implement this change.

  These findings point to the importance of digital forensic awareness initiatives as a means to reduce the reluctance to prioritize digital crimes and increase confidence levels when prosecuting crimes.

4.4 Cognitive Approaches

From the breadth of responses and the associated analysis, it is clear that the development of a digital forensic readiness plan extends beyond the realm of an organization. The corporate, industry and legislative environments each have properties that must be considered when developing a digital forensic readiness plan. These properties are presented in Figure 1.

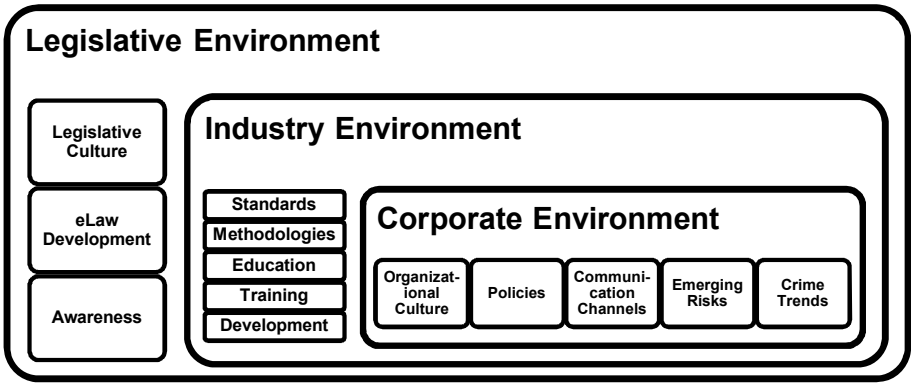


Figure 1. Cognitive approaches for digital forensic readiness planning.

The corporate environment has properties within an organization's control that must be aligned with and support the organization's digital forensic readiness plan. The corporate environment operates within the limitations of the industry environment, which, in turn, is influenced by the limitations of the legislative environment. A critical component of a digital forensic readiness plan is to establish a cooperation strategy that ensures that digital forensic cases can progress seamlessly from their inception in a corporate environment to their conclusion in a court of law (legislative environment), following relevant guidelines in the industry environment to ensure that the integrity of evidence is maintained.

5. Conclusions

This study has sought to investigate cognitive approaches that aid in developing digital forensic readiness plans. The study reveals that developing a digital forensic readiness plan is a task that involves many factors beyond the realm of a single organization. The factors are presented in the form of a conceptual model for organizations to use in process planning to achieve digital forensic readiness. By focusing on the lessons learned from experience, the study provides cognitive approaches to digital forensic readiness that can be used by individuals who wish to explore this topic further as well as by organizations that desire to enhance their ability to respond to security incidents while maintaining the integrity of evidence and keeping investigative costs low.

Our future research will examine digital forensic readiness as it relates to specific contexts such as mobile devices, wireless networks, public key infrastructures and cloud computing.

References

[1] ATLAS.ti Scientific Software Development, A world of data in your hand, Berlin, Germany (www.atlasti.com).

[2] M. Cohen, D. Bilby and G. Caronni, Distributed forensics and incident response in the enterprise, *Digital Investigation*, vol. 8(S), pp. S101–S110, 2011.

[3] J. Corbin and A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, Sage Publications, Thousand Oaks, California, 2008.

[4] J. Cresswell and V. Clark, *Designing and Conducting Mixed Methods Research*, Sage Publications, Thousand Oaks, California, 2011.

[5] S. Hoolachan and W. Glisson, Organizational handling of digital evidence, *Proceedings of the Conference on Digital Forensics, Security and Law*, pp. 33–44, 2010.

[6] P. Kanellis, E. Kiountouzis, N. Kolokotronics and D. Martakos (Eds.), *Digital Crime and Forensic Science in Cyberspace*, Idea Group Publishing, Hershey, Pennsylvania, 2006.

[7] Y. Manzano and A. Yasinsac, Policies to enhance computer and network forensics, *Proceedings of the Second Annual IEEE SMC Information Assurance Workshop*, pp. 289–295, 2001.

[8] M. Patton, *Qualitative Research and Evaluation Methods*, Sage Publications, Thousand Oaks, California, 2002.

[9] A. Pooe and L. Labuschagne, A conceptual model for digital forensic readiness, *Proceedings of the South African Information Security Conference*, 2012.

[10] Republic of South Africa, Electronic Communications and Transactions Act 2002, *Government Gazette*, vol. 446(2), no. 23708, August 2, 2002.

[11] J. Salmons, *Online Interviews in Real Time*, Sage Publications, Thousand Oaks, California, 2010.

[12] C. Teddlie and A. Tashakkori, *Foundations of Mixed Methods Research: Integrating Quantitative and Qualitative Approaches in the Social and Behavioral Sciences*, Sage Publications, Thousand Oaks, California, 2009.

Chapter 5

A HARMONIZED PROCESS MODEL FOR DIGITAL FORENSIC INVESTIGATION READINESS

Aleksandar Valjarevic and Hein Venter

Abstract Digital forensic readiness enables an organization to prepare itself to perform digital forensic investigations in an efficient and effective manner. The benefits include enhancing the admissibility of digital evidence, better utilization of resources and greater incident awareness. However, a harmonized process model for digital forensic readiness does not currently exist and, thus, there is a lack of effective and standardized implementations of digital forensic readiness within organizations. This paper presents a harmonized process model for digital forensic investigation readiness. The proposed model is holistic in nature and properly considers readiness and investigative activities along with the interface between the two types of activities.

Keywords: Digital forensic investigation readiness, process model

1. Introduction

We are living in an information society where we depend heavily on information systems and information technology. Therefore, we also depend on information systems security, specifically the confidentiality, integrity and availability of data, services and systems. These facts, combined with the increasing rate of information security incidents, make the field of digital forensics even more important.

Methods and process models for the digital forensic investigation process (DFIP) have been developed mostly by practitioners and forensic investigators based on their expertise and experience. The initial goal was to increase the effectiveness and efficiency of investigations, not necessarily to achieve harmonization or standardization. The same is true for the digital forensic investigation readiness process (DFIRP). There is

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 67–82, 2013.

no international standard that formalizes DFIP or DFIRP. However, at the time of writing this paper, an effort to standardize DFIP and DFIRP has been initiated by us within the International Standardization Organization (ISO). At this time, the standard is in its third working draft and is titled "ISO/IEC 27043 Information Technology – Security Techniques – Investigation Principles and Processes" [5]. Note that ISO/IEC 27043 is considering DFIRP as an integral part of DFIP and, ultimately, DFIRP should be contained within DFIP, i.e., within a single holistic DFIP model.

The focus of this paper is on a DFIRP implementation, not on the entire holistic implementation of DFIP as described in ISO/IEC 27043. The fundamental problem is that no harmonized DFIRP currently exists. This means that organizations do not have clear guidance on how to implement digital forensic investigation readiness.

Guidelines provided by a DFIRP model could enable organizations to better utilize their resources and achieve better results when implementing digital forensic readiness and when performing digital forensic investigations. The DFIRP model would help raise awareness and enhance training efforts related to digital forensic readiness and digital forensic investigations. Moreover, the model could enhance the quality of incidence response and investigations, and the admissibility of digital evidence.

2. Background

This section provides an overview of previous work related to digital forensic readiness, and models and processes used to achieve digital forensic readiness. This discussion is important because the existing digital forensic readiness models are inputs to the proposed harmonized model. Indeed, the proposed model attempts to harmonize existing models.

Digital forensics is the use of scientifically-derived and proven methods for the identification, collection, transport, storage, analysis, presentation and distribution and/or return and/or destruction of digital evidence derived from digital sources, while obtaining proper authorization for all actions, properly documenting all actions, interacting with the physical investigation, preserving evidence and the chain of evidence, for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping anticipate unauthorized actions that may disrupt operations [11]. Digital forensic readiness is the ability of an organization to maximize its potential to use digital evidence while minimizing the costs of an investigation [10].

Tan [10] has identified several factors that affect digital forensic readiness: how logging is done, what is logged, intrusion detection, digital forensic acquisition and digital evidence handling. Yasinsac and Manzano [7] have proposed six policy areas to facilitate digital forensic readiness: retaining information, planning the response, training, accelerating the investigation, preventing anonymous activities and protecting evidence. Wolfe-Wilson and Wolfe [12] emphasize the need for an organization to have procedures in place to preserve digital evidence in the event that a digital forensic investigation must be conducted.

Rowlingson [9] defines a number of goals for digital forensic readiness: gather admissible evidence legally and without interfering with business processes, gather evidence targeting the potential crimes and disputes that may adversely impact an organization, allow an investigation to proceed at a cost in proportion to the incident and ensure that the evidence makes a positive impact on the outcome of a legal action. Rowlingson's approach is closely related to our DFIRP model. His key activities when implementing digital forensic readiness are to: define the business scenarios that require digital evidence, identify the available sources and different types of potential evidence, determine the evidence collection requirements, establish a capability for securely gathering legally admissible evidence, establish a policy for secure storage and handling of potential evidence, implement monitoring to detect and deter major incidents, specify circumstances when escalation to a full investigation should be launched, train staff in incident awareness so that they understand their roles in the digital evidence process and the legal sensitivity of evidence, document an evidence-based case that describes the incident and its impact, and ensure legal review to facilitate actions in response to the incident.

Since the first Digital Forensic Research Workshop (DFRWS) [8], the need for a standard framework for digital forensics has been acknowledged by the digital forensics community. A framework for digital forensics must be flexible enough to support future technologies and different types of incidents. Therefore, it needs to be both simple and abstract. However, if it is too simple and too abstract, then it is difficult to create tool requirements and test procedures for the various phases [3].

Several researchers have proposed digital forensic models that include forensic readiness as a phase. However, to the best of our knowledge, no DFIRP model has as yet been proposed.

Carrier and Spafford [2] have proposed a digital investigation process model comprising seventeen phases divided into five groups, one of the groups focusing on forensic readiness; this group incorporates two phases, the operation readiness phase and the infrastructure readiness

phase. Mandia, *et al.* [6] have also proposed a digital investigation process model that includes a readiness phase, known as the pre-incident preparation phase. Beebe and Clark [1] have proposed a hierarchical, objectives-based framework for digital investigations, which includes a preparation phase; this phase encompasses activities designed to achieve digital forensic readiness.

3. Harmonized DFIRP Model

This section describes the proposed harmonized DFIRP model that is intended to provide guidance on implementing digital forensic readiness.

3.1 Aims and Policy

The harmonized DFIRP model has certain aims that are collectively drawn from previous work in the area [3, 7–10, 12]. In particular, the harmonized DFIRP model should:

- Maximize the potential use of digital evidence.
- Minimize the costs incurred in digital investigations.
- Minimize the interference to and prevent the interruption of business processes.
- Preserve or improve the current level of information security.

The fourth aim listed above was not identified in previous work. We believe that this aim is essential when implementing forensic readiness, and even more so when creating a DRIFP model. The first two aims concentrate on the efficiency of investigations and the third aim focuses on non-interference with business processes. Omitting the fourth aim could leave room for flaws in the overall information security status of an organization.

An example of such a flaw is when an organization, based on the first three aims, decides to collect logs from its information systems and maintain them at a central location. However, the organization does not implement security mechanisms to protect the data from compromise or dissemination. A holistic approach that applies information systems security mechanisms to forensic readiness is vital. In fact, forensic readiness should have built-in security features and security should not merely be an add-on.

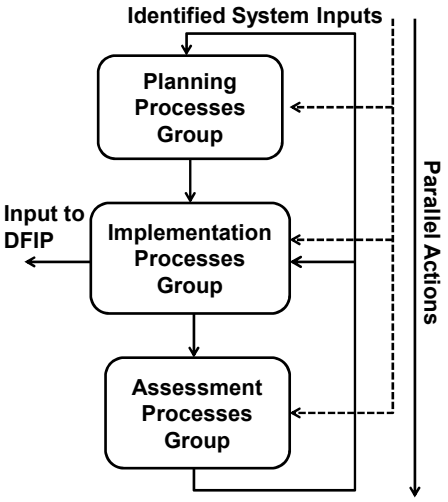


Figure 1. Harmonized DFIRP model: Process groups.

3.2 Model Description

This section describes the proposed DFIRP model in detail. Unlike related work [1] that uses the term "phases," we use the term "processes" in line with ISO terminology [4].

The harmonized model comprises three distinctive process groups: (i) planning processes group; (ii) implementation processes group; and (iii) assessment processes group. Figure 1 shows the three process groups.

The planning processes group includes all the model processes that are concerned with planning activities, including the scenario definition process, source identification process, planning pre-incident collection process, planning pre-incident analysis process, planning incident detection process and architecture definition process. These processes are shown in Figure 2.

The implementation processes group includes only the implementation processes and a link to the harmonized DFIP [11]. The implementation processes group includes the following processes: implementing architecture definition process, implementing pre-incident collection process, implementing pre-incident analysis process and implementing incident detection process. These processes, which are shown in Figure 2, are concerned with the implementation of the results of the planning processes.

The assessment processes group includes two processes, the assessment of implementation process and the implementation of assessment results process.

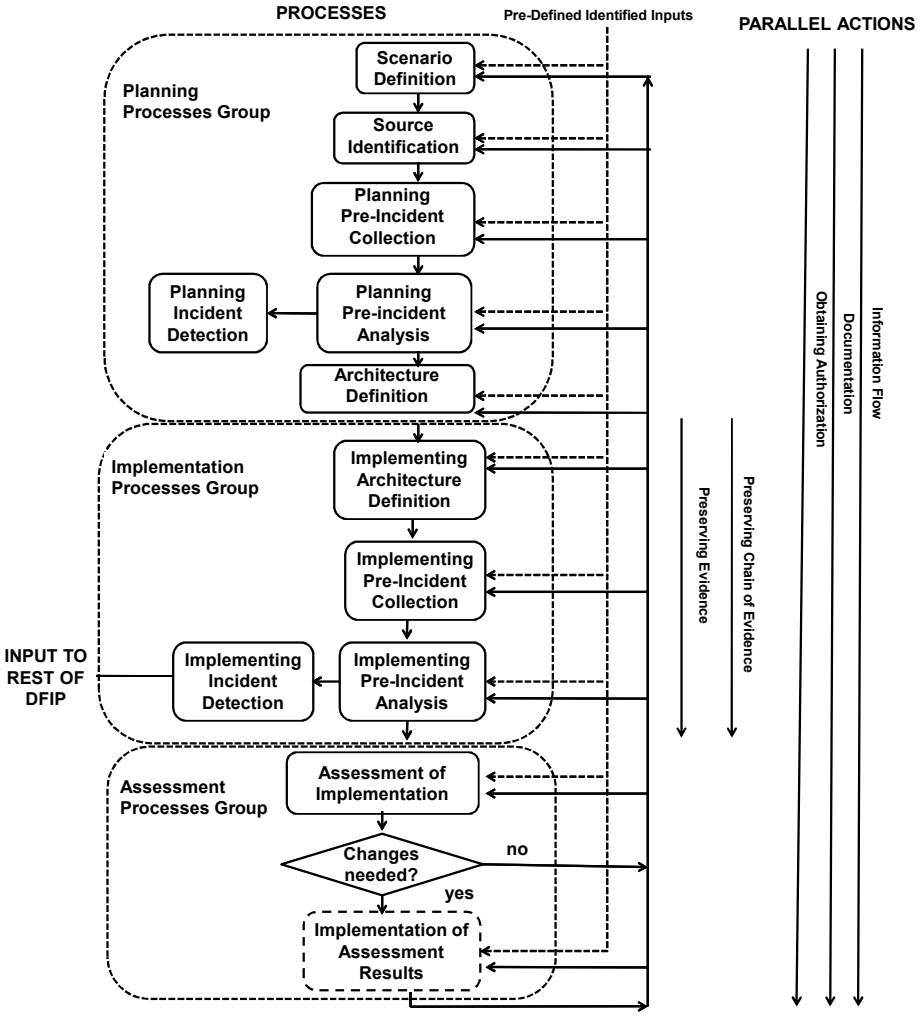


Figure 2. Harmonized DFIRP model.

The harmonized process model also introduces a novel addition to the process model – the concept of parallel actions. We define parallel actions as the principles that should be translated into actions throughout the DFIRP model. As shown in Figure 1, these actions run in parallel with model processes. The actions are described in more detail later in the paper.

The processes are defined at a high level in order to be used as a model for different types of digital forensic investigations. We do not attempt to prescribe the exact details of each process. Many different

types of digital forensic investigations exist, such as live forensics, cloud forensics, network forensics and mobile phone forensics. The detailed procedures for each process should be defined for each specific type of digital forensic investigation. However, defining all these procedures is outside the scope of this paper. The proposed model should, therefore, be used as an "umbrella model" for the various types of digital forensic investigations. The detailed procedures are left to be implemented by other standards and by the digital forensic community.

3.3 Model Application Environment

We use a hypothetical banking environment to illustrate the harmonized model and its component processes. We assume that the four aims of digital forensic readiness must be achieved at a particular branch of a bank.

The information system at the bank branch has the following components:

- Two workstations (personal computers) that run a banking application. Access to the workstation operating system and the banking application uses single-factor, password-based authentication. The banking application is a web-based, thin-client application. An application log is stored locally on the client and centrally on the bank server.
- Two digital cameras that record the activities of employees at the workstations.
- Two digital cameras that record activities at the branch.
- One networked video recorder that records data from all four cameras.
- One switch that connects the workstations, cameras and networked video recorder in a local-area network (LAN).
- A router that connects the branch LAN to the bank's wide-area network (WAN).

Certain policies are defined for the information system. The policies specify how the system should function:

- Networked video recorder data is backed up and taken off-site once a day.
- Clients are identified manually by inspecting their identity documents (IDs). Each transaction at a bank counter is authorized by a

banking client's signature. The signature must be compared with the signature on the banking client's ID. The ID with the banking client's signature is scanned and archived in a central location.

3.4 Model Processes

This section describes the processes in the harmonized model along with their inputs and outputs.

Scenario Definition Process. This process involves the examination of all scenarios that may require digital evidence. The input to this process includes all the information regarding system architecture, system technology (hardware and software), policies, procedures and business processes. The input also includes the DFIRP aims. We refer to this set of inputs as "pre-known system inputs."

Similar inputs exist for all the other processes in the model. For example, the pre-known system inputs may include the network topology, specifications of models and hardware components, specifications of firmware, operating systems and applications for each piece of hardware, information security policies governing system use and the descriptions of the business use of the system to which the model is applied.

The output of this process is a set of scenario definitions. The scenarios may correspond to information security incidents such as the unauthorized use of resources. They may also correspond to events that require digital forensic investigations, such as the use of a computer to distribute child pornography.

During the scenario definition process, a risk assessment should be performed for each scenario separately. A risk assessment helps identify the possible threats, vulnerabilities and related scenarios where digital evidence might be required. Based on the assessed risk from certain threats, vulnerabilities or scenarios, the later processes can be used to better decide on the measures necessary to achieve forensic readiness, taking into account the risk levels, costs and benefits of the possible measures in order to reduce the identified risk. For example, a better decision can be made about the need to centrally collect and process all system log data in order to improve digital forensic readiness. In addition, the risk assessment would help determine the protection mechanisms needed for the centralized storage of log data, such as firewalls, link encryption, storage data encryption and change tracking.

An example scenario is the misuse of the banking application, where the application is used to steal credit card information. Another scenario is a complaint from a client claiming that he did not withdraw money from his account at the branch at a certain date.

The scenario definition process is a logical start for the DFIRP implementation because proper scenario analysis lays the foundation for all subsequent processes. After this initial process, it is necessary to specify all the possible sources of digital evidence based on the defined scenarios.

Source Identification Process. During this process, it is necessary to identify all the possible sources of digital evidence in the information system. Note that, for reasons of simplicity, inputs are represented as single arrows in Figure 1. The output of the process is the possible sources of evidence, e.g., registry files, temporary Internet files, email archives and application logs.

Some of the identified sources of evidence might not be available. For example, access logging may be not implemented in the information system. In such cases, methods for making the identified sources available and the use of alternative sources should be explored.

In our hypothetical information system, the possible sources of evidence based on the identified scenarios are:

- Data from the two workstations, especially banking application logs, temporary Internet files, text editor logs, email stored on the workstations, traces of deleted files and traces of modified files.
- Banking application logs stored at the bank's data center.
- Banking transaction data, such as signed transaction documents, stored at a central location.
- Video recordings from the networked video recorder.
- Backed-up data from the networked video recorder.

After the possible evidentiary sources have been identified, it is necessary to specify how these sources should be handled. This is accomplished using the planning pre-incident collection process and the planning pre-incident analysis process.

Planning Pre-Incident Collection Process. During this process, procedures are defined for pre-incident collection, storage and manipulation of data that represents possible digital evidence. Note that the data collection period is determined based on a risk assessment. For example, this could mean determining how often an organization would save the application log to a central repository to ensure the integrity of log data in the event that the application is compromised. The collection, storage and manipulation of data must conform to digital forensic principles

(e.g., chain of custody and evidence preservation) so that the digital evidence is admissible in a court of law. Also, the data retention period should be determined based on two factors: (i) risk assessment; and (ii) previous experience regarding incident detection, data quantity, network capacity and other matters that could influence the cost or efficiency of the process.

In the case of the hypothetical banking information system, the collection procedures for possible sources are:

- Collection of images of workstations is to be performed once a week. Images are to be stored off-site, securely and safely to preserve evidence. They are to be retained for a period of one year.
- Collection of banking application logs stored locally, temporary Internet files and text editors logs is to be performed daily. The data is to be sent via the network connection to the central repository. The data is to be retained for a period of five years.
- Deleting emails from the email server is forbidden. The emails are to be retained for a longer period of time, depending on the prevailing laws and regulations.
- Banking application logs and banking transactions data stored centrally are to be backed-up daily. Backed-up data is to be retained for a period of ten years.
- Video recordings from the networked video recorder are to be streamed to a central networked video recorder and retained for a period of two years.
- Backed-up data from the networked video recorder is to be retained for a period of two years.
- An intrusion prevention system is to be used to collect LAN network traffic data and traffic data to/from the WAN. This data is to be stored at a central location and retained for a period of one month.

Planning Pre-Incident Analysis Process. This process defines procedures for pre-incident analysis of data representing possible digital evidence. The aim of the analysis is to enable incident detection. Therefore, the procedures defined in this process must include exact information on how incidents are detected and the behavior that constitutes each incident.

The tasks of data analysis and incident detection are often outside the scope of target information systems (information systems that might come under a digital forensic investigation). Therefore, we recommend that this process defines an interface between the information system and a monitoring system that analyzes data in order to detect incidents.

In the case of our hypothetical banking information system, it would be necessary to have custom scripts or commercial software (e.g., for change tracking, intrusion detection and business intelligence) to analyze the information collected (both locally and centrally) in order to detect anomalies and possible incidents. Information security best practices should be taken into consideration as well as bank business processes and policies.

Planning Incident Detection Process. Since the main goal of the planning pre-incident analysis process is to enable incident detection, the next logical process is the planning incident detection process. The output of this process includes the actions to be performed after an incident is detected, especially collecting the information to be passed to the digital forensic investigation process. The information should also include pre-known system inputs, results from all DFIRP processes and data gathered and generated during the implementation process.

In the case of our hypothetical banking information system, when an incident is detected via pre-incident analysis, the information gathered by relevant software (e.g., for change tracking, intrusion detection and business intelligence) should be automatically sent to the bank's central information system and should trigger incident response activities.

Architecture Definition Process. This process involves the definition of an information system architecture for the information system that is to be forensic ready. The process draws on the results of all the previous processes. The process is introduced in order to implement better forensic readiness by taking into account all relevant matters when redefining the system architecture. The aim is to customize the system architecture to accommodate the four DFIRP aims.

In the case of our hypothetical information system, the architecture definition would include decisions to store no application data locally, to introduce automated document reading and biometric identification to verify customer identity, and to introduce three-factor authentication for workstations and applications (e.g., PINs, biometrics and smart cards).

Implementation Processes Group. This group implements the results of all the previous processes. Although the processes compris-

ing this group are distinctive, they are presented in one section, unlike the other processes that are presented separately. This is because all the processes in the implementation processes group are concerned with the implementation of results from the planning processes group. All these processes represent the implementation of technical or non-technical measures defined in the other processes.

In practice, the measures defined in the architecture definition process should be implemented. This is followed by the pre-incident collection, pre-incident analysis and incident detection processes.

A clear difference exists between the processes in the implementation processes group (Figure 2) and the processes in the planning processes group. The difference is that, for example, the process listed as implementing pre-incident collection in the planning processes group is tasked with defining what data is collected and how it is collected; on the other hand, the implementing pre-incident collection process of the implementation processes group is tasked with implementing the results of the implementing pre-incident collection process, including digital evidence collection.

It is important that the roles of the various people in the system are considered. People represent users. However, people are also custodians and owners of information system components. The procedures must include relevant information for all the people involved with the system. Also, training and awareness sessions must be conducted for all people involved with the information system.

The output of the implementation processes group is an information system that is finally forensically ready. This process represents an interface to the DFIP; in fact, it straddles the tasks of readiness and investigation.

Assessment of Implementation Process. After forensic readiness been implemented, it is necessary to start the assessment process. This process examines the results of the implementation of forensic readiness to determine if it conforms to the DFIRP aims.

During this process, a legal revision should be carried out for all procedures, measures and architectures defined when implementing the model. The revision should show whether or not there is conformity with the legal environment and digital forensic principles in order to ensure evidence admissibility.

In the case of our hypothetical banking information system, the assessment process would take the form of an internal audit or external audit. The goals of the audit would be to check if the implementation

conforms to the results of the planning processes group and the four DFIRP aims.

Implementation of Assessment Results Process. This process is concerned with implementing the conclusions from the previous process. The process is optional because it is possible that no actions are needed based on the results of the assessment of implementation process.

During this process, it is necessary to decide on recommendations for changes in one or more of the previous processes. The main decision here is whether to go back to one of the planning processes or to go back to an implementation process based on the results of the implementation assessment process.

3.5 Parallel Actions

This section discusses the actions that must run in parallel with the processes. The parallel actions are defined as the principles that should be translated into actions in the DFIRP. Examples are the principle that evidentiary integrity must be preserved throughout the process and that the chain of evidence must be preserved. These principles are found in existing DFIP models [9–11].

The parallel actions in the DFIRP model are: preserving the chain of evidence, preserving evidence, defining information flow, documentation and obtaining authorization (Figure 2). These actions are implementations of well-established principles of digital forensics.

The actions run in parallel with all the other processes to ensure the admissibility of digital evidence. Parallel actions also enhance the efficiency of an investigation. Some actions defined by other researchers, such as obtaining authorization and preparing documentation, run across several processes.

Preserving the Chain of Evidence. All legal requirements must be complied with and all actions involving digital evidence must be properly documented in order to preserve the chain of evidence and the integrity of evidence. This principle must be followed throughout the DFIRP.

Preserving Evidence. The integrity of the original digital evidence must be preserved. This is achieved using strict procedures from the time that the incident is detected to the time that the investigation is closed. The procedures must ensure that the original evidence is not changed and, even more importantly, they must guarantee that no opportunity for evidence tampering arises during the entire investigation.

Information Flow. It is important to identify and describe all information flows so that they can be supported and protected. An example information flow is the exchange of digital evidence between two investigators. This information flow could be protected using a public key infrastructure to preserve confidentiality, timestamping to identify the different investigators, and authenticating the evidence to protect its integrity.

A defined information flow should exist between each of the processes and between the various stakeholders, including investigators, managers and external organizations. For example, there should be defined information flows between investigators and managers of the information system being investigated, including obtaining authorizations from the managers (system owners or system custodians), and informing managers about security incidents, their possible consequences and the required actions.

Documentation. Every action performed should be documented to preserve the chain of evidence, and to increase efficiency, resource utilization and the likelihood of a successful investigation. This would, for example, include documenting how the information obtained during pre-incident collection has been stored and processed, along with all the persons who have had access to the information.

Obtaining Authorization. Proper authorization should be obtained for every action that is performed. Depending on the action, the authorization may have to come from government entities, system owners, system custodians and/or principals.

4. Conclusions

The proposed model harmonizes existing efforts related to DFIRP models. It has a broader scope than digital forensic readiness processes and existing digital forensic process models. The broader scope is manifested by the definition of additional processes such as the planning pre-incident data analysis process, architecture definition process and assessment processes.

The processes in the harmonized model are well-defined in terms of scope and functions. The processes deal with all the matters covered by existing models as well as matters that are outside the scope of existing models, such as customizing the architecture definition of a target information system to achieve the digital forensic readiness goals. In fact, customizing the architecture definition is a novel concept that contributes to a more holistic approach to digital forensic readiness.

Another novel concept is the incorporation of actions based on digital forensic principles that are performed in parallel with processes in the harmonized model. These parallel actions enhance the efficiency of investigations and ensure the admissibility of digital evidence.

Using the harmonized DFIRP model provides several benefits. These include improved admissibility of digital evidence, reduced human errors and omissions during the DFIRP, and better resource utilization when implementing digital forensic readiness and conducting digital forensic investigations. Also, the harmonized DFIRP model can improve the overall security of an information system by raising awareness about specific incidents and alternative scenarios.

Our future work will apply the harmonized model to a functional system and measure its conformance with the four DFIRP aims, both before and after the implementation. Future work will also involve defining an interface between the harmonized DFIRP model and a DFIP model to achieve a holistic and harmonized DFIP model. Initial work related to these topics and others is discussed in [5].

References

[1] N. Beebe and J. Clark, A hierarchical, objectives-based framework for the digital investigations process, *Digital Investigation*, vol. 2(2), pp. 146–166, 2005.

[2] B. Carrier and E. Spafford, Getting physical with the digital investigation process, *International Journal of Digital Evidence*, vol. 2(2), 2003.

[3] B. Carrier and E. Spafford, An event-based digital forensic investigation framework, *Proceedings of the Fourth Digital Forensics Research Workshop*, 2004.

[4] International Standards Organization and International Electrotechnical Commission, ISO/IEC 12207, Systems and Software Engineering – Software Life Cycle Processes, Geneva, Switzerland, 2008.

[5] International Standards Organization and International Electrotechnical Commission, ISO/IEC 27043 – Information Technology – Security Techniques – Digital Evidence Investigation Principles and Processes (Draft), Geneva, Switzerland, 2012.

[6] K. Mandia, C. Prosise and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.

[7] Y. Manzano and A. Yasinsac, Policies to enhance computer and network forensics, *Proceedings of the Second Annual IEEE SMC Information Assurance Workshop*, pp. 289–295, 2001.

[8] G. Palmer, A Road Map for Digital Forensic Research, DFRWS Technical Report DTR-T001-01 Final, Digital Forensic Research Workshop, Utica, New York (`www.dfrws.org/2001/dfrws-rm-final.pdf`), 2001.

[9] R. Rowlingson, A ten step process for forensic readiness, *International Journal of Digital Evidence*, vol. 2(3), 2004.

[10] J. Tan, Forensic readiness: Strategic thinking on incident response, presented at the *Second Annual CanSecWest Conference*, 2001.

[11] A. Valjarevic and H. Venter, Harmonized digital forensic investigation process model, *Proceedings of Eleventh Annual South African Information Security Conference*, 2012.

[12] J. Wolfe-Wilson and H. Wolfe, Management strategies for implementing forensic security measures, *Information Security Technical Report*, vol. 8(2), pp. 55–64, 2003.

Chapter 6

EVALUATION OF THE SEMI-AUTOMATED CRIME-SPECIFIC DIGITAL TRIAGE PROCESS MODEL

Gary Cantrell and David Dampier

Abstract The digital forensic process as traditionally laid out is very time intensive – it begins with the collection, duplication and authentication of every piece of digital media prior to examination. Digital triage, a process that takes place prior to this standard methodology, can be used to speed up the process and provide valuable intelligence without subjecting digital evidence to a full examination. This quick intelligence can be used in the field for search and seizure guidance, in the office to determine if media is worth sending out for an examination, or in the laboratory to prioritize cases for analysis. For digital triage to become accepted by the forensic community, it must be modeled, tested and peer reviewed, but there have been very few attempts to model digital triage. This work describes the evaluation of the Semi-Automated Crime-Specific Digital Triage Process Model, and presents the results of five experimental trials.

Keywords: Digital triage, process model, evaluation

1. Introduction

Digital forensics involves the post-event processing of digital media for artifacts of interest. An event in this case means a crime against a computer, a crime where a computer was a tool, or a crime where the computer was incidental [16]. The artifacts correspond to digital data that can serve as intelligence for a case under investigation or serve as evidence in a court of law. Since these artifacts are to be used in a court of law, they must be gathered using proven, forensically-sound methodologies. At this time, these methodologies are typically standard operating

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 83–98, 2013.

procedures that have been created independently by the organizations involved in forensic investigations.

Digital triage is a pre-digital-forensic process performed on a live or dead system. For a live system, digital triage is typically performed to extract information that would be lost when the system is powered down. This information could be stored in volatile memory or on an internal drive in an encrypted format. For a dead system, digital triage is typically performed to gather quick information for intelligence purposes. The intelligence can have many uses, including, but not limited to, determining if an examination is warranted, providing plea bargaining assistance, focusing examination efforts and guiding search and seizure efforts.

Digital triage is a pre-digital-forensic process because it is carried out prior to the accepted digital forensic practice of imaging and authenticating each piece of media before examining it. Digital triage thus examines the original evidence whereas an accepted digital forensic methodology examines an image (forensic copy) of the original evidence.

Several models have been proposed to formalize the discipline of digital forensics and to transform *ad hoc* processes into tested and proven methodologies [2–4, 6, 7]. Although some of these models mention the need for a pre-examination process such as digital triage, none of them include digital triage as a detailed phase. Moreover, very few standalone models have been proposed for digital triage [5, 18]. Thus, digital triage is mostly an unmodeled component in existing process models.

This paper discusses the evaluation of the Semi-Automated Crime-Specific Digital Triage Process Model [5]. The model was created to decrease the time taken to perform a digital triage assessment by at least 50% compared with an *ad hoc* process, but without reducing the accuracy. The model and its implementation are described in detail, and the results of five experimental trials are presented.

2. Process Model

The Semi-Automated Crime-Specific Digital Triage Process Model is designed to be used by novices. It incorporates ongoing concerns regarding pre-digital-forensic processes and can be specialized for different classes of crimes. The most significant contribution of the model is its automated phases (shaded rectangles in Figure 1). Planning and readiness is an on-going phase that occurs pre-event while preservation is an umbrella activity that is performed during all the phases. The remaining phases are presented in a linear fashion. The dotted line in the left-hand side of Figure 1 represents information flow from the computer

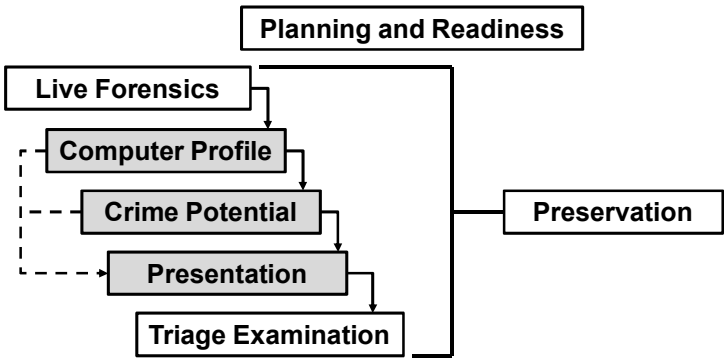


Figure 1. Semi-Automated Crime-Specific Digital Triage Process Model.

profile and crime potential phase to the presentation phase, where the raw information is transformed into usable information for the digital triage examiner.

The planning and readiness phase is an ongoing phase involving the preparation and education of staff, and the continual upgrading of equipment. Including this phase in a digital triage process is important because it ensures the continual testing of triage tools and efforts on the part of the triage examiner to keep abreast of the tools.

Live forensics involves the acquisition and analysis of volatile memory or the analysis of static data on a live machine. It is included as an optional phase based on need and expertise. Digital evidence should be gathered in the order of volatility to prevent the loss of data of evidentiary value [1, 11]. Thus, this step must come before anything else in any digital forensic or pre-digital forensic process. However, it is often skipped in investigations where the volatile memory is ignored.

Computer profiling is the first phase of an automated process in a digital triage process model. In the Field Triage Process Model [18], the intent is to learn about the users of a system by targeting the user profiles on a computer. The Five Minute Forensic Technique [12] incorporates a similar analysis, but the information is used to categorize the users as occasional users, Internet/chat users, office worker users, experienced users and hacker users. On the other hand, the Semi-Automated Crime-Specific Digital Triage Process Model suggests profiling a computer as a whole and dividing the data by volume and by user directory.

The computer profiling phase is the same for every piece of media examined. In contrast, the crime potential phase contains only the components that are dependent on a specific crime class. The phase attempts to guide the triage examiner by raising the red flags that should be con-

sidered for a specific crime class. Information is also gathered during the computer profiling phase using word searches and known file searches. Although this is listed as a separate phase, it could run concurrently with the computer profiling phase to save time.

During the presentation phase, information from the computer profile and crime potential phases are incorporated in a report that can lead the digital triage examiner to artifacts of interest or help the examiner determine how the evidence should be prioritized. The results are interpreted and applied according to need.

During the triage examination phase, the evidence is viewed in a forensically-safe manner according to the guidance provided by the presentation phase. This phase is optional based on need. If the presentation phase has produced enough information, there may be no need for further examination. The triage examination phase corresponds to the *ad hoc* method that is typically used by law enforcement.

Interested readers are referred to Cantrell, *et al.* [5] for additional details about the Semi-Automated Crime-Specific Digital Triage Process Model and its various phases.

3. Model Implementation

In an effort to facilitate model evaluation, we developed a series of scripts for the automated phases of the model: computer profiling, crime potential and presentation. The final product, which comprises open source and custom tools written in Perl, is called the Fast Modular Profiling Utility (FMPU). FMPU quickly gathers useful information to create a profile of a computer. As FMPU creates this profile, the information is monitored for keywords to assist in the determination of the crime potential. FMPU then presents the information to the user as a main report and a red flag alert report, both of them in an HTML report format.

It was decided to employ a modular design for the digital triage tool. The modular design allows for easy expansion and simple customization, and facilitates the incorporation of commands and tools.

Seven steps are involved in the execution of the tool:

- **Step 1:** Main Module accepts the report name and location as input.
- **Step 2:** Main Module writes the HTML header.
- **Step 3:** Main Module passes control to Module 1.
- **Step 4:** Module 1 extracts information.

- **Step 5:** Module 1 formats information as text, HTML table or separate HTML pages.
- **Step 6:** Module 1 appends text, HTML table or HTML links to the appropriate report.
- **Step 7:** Main Module creates the HTML footer to close the report.

Steps 3 through 6 are executed for each data item extracted. A series of scripts incorporating open source and original code were created to implement the desired functionality. Small amounts of information are added directly to the report. Larger sub-reports are written as separate files and linked to the main report or alert report.

FMPU gathers the following information to create a computer profile:

- **File System Information:**
  - Physical/logical disk layout
  - Sector allocation
  - File system types and locations
- **File Classification:**
  - File type report for each user directory
  - File type report per volume
- **Application Information:**
  - Usernames in the system
  - Web browser history
  - Windows registry data

The file system information that is gathered includes the physical disks attached to the computer being examined, the logical volumes available for mounting, the sector layout of the system, and the file system label of each volume. Viewing the physical and logical layouts of the system enables the digital triage examiner to quickly determine the amount of data on the system and the organization of each disk.

FMPU gathers file statistics from the system based on file type. This enables the digital triage examiner to quickly create a theory regarding the use of the system. For example, a media machine is likely to have a large variety of video and sound files. A corporate machine would be more likely to have documents and spreadsheets. Unfortunately, it is difficult to conjecture what "normal users" would have on their machines.

Therefore, the classification is left to the experience of the digital triage examiner and the specific template that is employed.

Application information is information that is collected about the user by an application, including the operating system, potentially without user consent. When building a computer profile, FMPU gathers usernames, web browser history and Windows registry information. Usernames are collected by pulling the user directories as listed on the system. This can provide an indication of the number of users on the system along with their identities. However, the digital triage examiner must recognize that there is no easy way to tell exactly who is using or has used a given account. Also, user directories may be placed in non-standard locations, which complicates the task. However, advanced FMPU users could edit the configuration file to specify the user directory locations that can be determined through the examination of system files.

In order to speed up future web browsing, most browsers maintain records of the sites visited by users. This history is preserved unless a user specifically changes the default setting. The first version of FMPU performed web history analysis only for Internet Explorer. This was done because Internet Explorer is arguably the most popular web browser. Internet Explorer stores its web history in `index.dat` files; the structure of these files is well-researched and documented [14, 15, 17].

The goal of FMPU is not to present all possible data. Rather, FMPU is designed to selectively collect items that are of the most interest to the digital examiner and to present the information in a useful manner. For example, in addition to listing URLs, it counts the number of times each domain was visited. The final listing is then sorted by the number of visits and sent as output. The raw output used to create this list is also included in the report in case the triage examiner needs more detail about specific links.

The Windows registry is a database of settings that provides important information to a digital forensic examiner [9, 10]. The format of these settings is not user friendly, and the settings are typically edited by software utilities rather than directly by users. FMPU uses the open-source RegRipper tool [8] to access registry data that it includes in the final report. The information extracted by RegRipper can be adjusted by the digital triage examiner using FMPU. In particular, FMPU collects login information, web history, instant messenger information, connected USB devices and shutdown information.

The final report is provided in the form of easy-to-navigate HTML pages. HTML is a common format for digital forensic reports. The final report is separated into two reports, a main report with all the data and an alert report containing data identified during the crime potential

phase. Users have the option to populate an input list of red flag words prior to running FMPU to facilitate activities during the crime profiling phase. This list is used to identify data that is of special interest. File system information is not changed during this phase; however, the files are classified. During the classification process, filenames that contain any keywords or known filenames are identified and flagged for inclusion in the alert report. Application information is also filtered. As the application information is gathered, it is monitored for keywords and known filenames, and anything identified is also included in the alert report.

4. Evaluation Results

This section describes the experimental procedure and the results of the evaluation of the Semi-Automated Digital Triage Process Model.

The subjects involved in the evaluation were divided into two groups, a validation group and a testing group. The validation group comprised qualified examiners while the testing group consisted of college students who were taking digital forensics courses. The validation group conducted qualitative testing of FMPU on real evidence. On the other hand, the testing group performed quantitative analyses to determine if using FMPU yielded a 50% decrease in processing time without any loss of accuracy compared with an *ad hoc* procedure. Only the validation group was allowed to work with real evidence due to restrictions on accessing real case data. Artificial data sets were created for use by the testing group.

4.1 Validation Results

The validation group consisted of four active digital forensic examiners. These subjects were provided with FMPU and a minimal set of instructions on how to use it. Prior to running the tool on real evidence, they were given a demonstration of FMPU on a test drive and an explanation of the results. The subjects were also asked to list five data elements that would have been useful to know prior to their original examination of the test data. Some of the items listed were large amounts of images, evidence of peer-to-peer sharing and documents with names of interest. After finalizing their lists, the subjects were asked to run the tool and examine the report.

The results of the validation testing supported FMPU. Table 1 shows the numbers of items listed and the numbers of items found by the subjects. All the subjects found at least one item that was predicted

Table 1. Validation group results.

Subject	Case Type	Items Listed	Items Found
1	Child Pornography	5	4
2	Child Pornography	4	3
3	Slander	4	1
4	Child Pornography	4	3

after hearing a description of the tool. Also, all the subjects stated that they found several items they had not predicted, but that were beneficial.

Table 2. Validation group responses.

Rating	Agree	Somewhat Agree	Do Not Agree	Not Applicable
Decrease Triage Time	2	2	0	0
Decrease Exam Time	3	1	0	0
Allow Case Prioritization	4	0	0	0

Table 2 presents the general opinions of the tool as provided by the validation subjects. The majority of the subjects believed that the tool would decrease triage time, examination time and allow for case prioritization. No validation subjects disagreed or felt that the tool would not be applicable.

The results also show that FMPU works on real-world evidence. Every subject found items that were anticipated as well as items that had not been listed before running FMPU. Even in the tests with limited results, it was noted that there was also very little found in the original examinations. During the more successful attempts, the subjects noted that, while items identified during the original examination were found, they were found much quicker using FMPU. These results support the use of FMPU on real evidence and facilitated the testing procedures conducted using artificial evidence sets.

4.2 Testing Results

The testing group was divided into experimental and control groups. The assignment of subjects to the experimental and control groups was done randomly. Experimental group subjects used FMPU while control group subjects used an *ad hoc* digital triage procedure.

The tests were quantitative in nature. As mentioned earlier, the testing group subjects were students with basic knowledge of the digital

forensic process, not active examiners. This substitution is acceptable because digital triage is conducted by individuals with varying levels of expertise, and it should be possible for novices to use a digital triage tool.

Subjects in both the experimental and control groups were given lectures on digital triage. The subjects performed short exercises or viewed demonstrations associated with each topic as part of the lectures. The lectures included the following topics:

- Using Linux boot environments (CD and USB).
- Mounting and viewing attached disks in Linux.
- Finding, listing and sorting files in Linux.
- Using the `strings` command to strip and view file content.
- Using the RegRipper utility.
- Using the Linux `file` command.

These concepts are the core of FMPU, and covering these topics ensured that most subjects could perform digital triage at a basic level before beginning the experiments. The subjects were encouraged to take notes and record the individual commands. They were allowed to bring these and any other materials they desired to the testing.

The subjects were further divided into five trials based on their levels of expertise and the testing locations. Each test subject was given the timed task of classifying three drives attached to the test machine by crime category – nothing of interest, murder scenario or child pornography (kitten pictures and phrases were used instead of real pornography). The experimental group performed this task using FMPU while the control group used an *ad hoc* procedure. The classification of these drives by crime category assesses the ability of the digital triage examiner to make decisions about digital evidence. Subjects were also asked for the confidence level regarding their selections – very confident, somewhat confident or complete guess. This allowed for outlier evaluation for subjects who gave up or simply guessed the answers.

Subjects in Trials 1 and 2 comprised law enforcement officers who were attending training courses at the Mississippi State National Forensics Training Center. Trial 1 had four subjects and Trial 2 had eight subjects. The subjects in these two trials had intermediate level expertise – they did not have a great deal of digital forensics training, but they more than made up for it with real-world investigation experience.

The subjects in Trials 3 through 5 were college students who were enrolled in digital forensics courses at Dixie State University in St. George, Utah. The subjects were given the same lectures and demonstrations as those in Trials 1 and 2, but the testing was carried out during scheduled time slots over a seven-day period. The expertise levels of the subjects in Trials 3–5 were determined based on their academic status. Trial 3 subjects were enrolled in 3000 and 4000 level classes and were classified as "advanced." Trial 4 subjects were enrolled in 2000 level courses and were classified as "intermediate." Trial 5 subjects were enrolled in 1000 level courses and were classified as "novice."

Note that, although Trial 1 is included in the testing results, it was not used in computing of the final mean times. It was, however, used in the final computation of accuracy. In addition to helping evaluate the primary goals of the research, Trial 1 was conducted in order to assess the effectiveness of the test data sets, evaluate the use of the tools on three test data sets at once, and refine the presentation materials given to the subjects prior to testing.

There was little variation in the times for the experimental and control group subjects in Trial 1, and there was only a decrease of 19% in the average times in favor of the experimental group. There was, however, an increase in average accuracy from 1.5 out of 3 to 2 out of 3 in favor of the experimental group. This can be attributed to the small size of the Trial 1 population. However, it was questioned if the tool itself could have been a contributing factor.

The test procedure and tool function were closely examined after Trial 1 was completed. The computer used for testing connected all three test sets at the same time to make the testing easier. However, this created considerable wait time before the subjects could begin the examination process.

FMPU was used to examine each volume in turn and to produce the report separated by physical disk, logical volume and user when possible. Each of the three test sets took approximately 4.5 minutes to process, resulting in more than 15 minutes of total processing time. Although 4.5 minutes does not seem like a lot of time, the 15 minutes taken to perform a combined analysis can stretch the patience of test subjects. More importantly, the time taken may be too large to provide an advantage over the *ad hoc* procedure used by the control group.

An analysis of FMPU revealed that 80% of the processing time was involved in file classification. In an effort to determine the use of a volume, FMPU classifies all the files by file type and creates a summary of the information. The first iteration performs this identification based on the file signature, a common digital forensic technique. An

Table 3. Experimental results by expertise.

Group	Expertise	Average Time (Minutes)	Standard Deviation	Accuracy
Experimental	Novice	19.00	14.85	100%
Experimental	Intermediate	16.60	7.09	100%
Experimental	Expert	10.89	9.77	93%
Control	Novice	33.33	20.59	44%
Control	Intermediate	32.00	18.09	40%
Control	Expert	37.14	19.55	70%

alternative file identification technique is to use the file extension. This type of identification is less accurate because a user or an application can intentionally rename a file extension in an attempt to obscure data. However, this is acceptable in digital triage because it is about collecting quick intelligence, not performing analysis.

During Trial 1, it was also observed that the file classification results did not seem to be as vital as the more specific application information that was gathered. With this in mind, prior to running Trial 2, the tool was set to perform file classification based on file extension instead of the first-byte signature. This change decreased the total processing time for the three drives from fifteen minutes to just four minutes. Therefore, the remaining trials were conducted using file extension identification instead of first-byte signature identification. The type of classification desired is easily set using the FMPU configuration file.

Trial 1 helped validate the quality of the test sets. Two users were able to identify all the test sets correctly with at least some confidence. Also, even with the delay imposed by file identification, crime classification could be completed within a reasonable amount of time. The accuracy results had a wide spread. Some subjects were completely correct, some were partially correct, and a few were totally wrong. These observations were also supported by the subsequent trials.

The independent variables in the tests are the presence of the FMPU report and level of expertise. The dependent variables are time and accuracy. The most apparent result in Table 3 is the difference in accuracy between the experimental and control groups. Most test subjects in the experimental group had 100% accuracy on average while the control group subjects had less than 50%, with the exception of the expert subjects in the control group (but their accuracy ratings are less than the lowest accuracy ratings in the experimental group).

The evaluation of processing time is more complex. It is not unreasonable to believe that the FMPU report would provide a greater benefit to novices than to intermediate users and a greater benefit to intermediate users than to experts. Thus, the subjects were divided into trials based on their expertise level and testing location. This division was made in an effort to explore this predicted effect. However, the results in Table 3 do not support this prediction. The expert subjects had a mean time decrease of 71% between the experimental and control groups. In the case of novice subjects, the mean time decrease was only 43%.

It is reasonable to assume that a greater level of digital forensics expertise would facilitate quicker and easier use of the tool. The average time for the experimental group subjects supports this assumption. However, this assumption does not appear to hold for the control group subjects.

We suspect that the lack of predictable responses in the control group is due to a higher degree of random chance and varying levels of expertise with the Linux environment used in the tests. The control group subjects with Linux experience were likely more confident in their results, but the likelihood of success and speed were dependent on if and how quickly they found the data that allowed them to make decisions. After all, finding the pertinent files can be a matter of chance with an *ad hoc* approach.

Another element of randomness was introduced by the subjects who guessed the results. For example, Subjects 2 and 3 indicated that their responses were complete guesses, but they took 45 minutes to complete the task. On the other hand, Subjects 4, 5 and 6, who also guessed, did so after only thirteen minutes. This randomness does not invalidate the results, it just makes the analysis more complex.

The effect of expertise level on accuracy and mean time was investigated further. We suspected that the expertise level had little or no significant effect on time or accuracy of the group as a whole because the expertise level classification was one dimensional. The experimental group subjects all followed the same process with assistance from FMPU, thereby normalizing their times. However, the control group subjects each approached the task in an *ad hoc* manner, leading to chance playing a greater role in the outcome. This leads us to believe the best evaluation metrics would be for the entire group combined or to limit the comparison to the subjects who correctly classified all three drives, which would serve to reduce the effect of the outliers who guessed and gave up at random times.

A multivariate analysis of variance (MANOVA) test was used to investigate the interaction between the independent variables (expertise and presence of the FMPU report) and the dependent variables (time

Table 4. MANOVA univariate test results.

Independent Variable	Dependent Variable	Degrees of Freedom	F-Value	p-Value
Expertise	Time	2, 44	0.131	0.878
Expertise	Accuracy	2, 44	0.863	0.429
FMPU Report	Time	1, 44	18.40	0.000
FMPU Report	Accuracy	1, 44	18.99	0.000

and accuracy). A MANOVA test compares the means of several groups to determine the statistical significance of the groups. It addresses the interaction significance between the dependent and independent variables. MANOVA is typically used when there is the possibility of noise caused by the interaction of the variables. In this circumstance, noise can be attributed to random chance in the control group and varying digital forensics and/or Linux expertise. MANOVA reports the different interactions of the variables as separate univariate results as part of the primary multivariate analysis [13].

Table 4 presents the MANOVA univariate test results. A p-value of 0.05 was used to determine if the null hypothesis is rejected or fails to be rejected. The null hypothesis in this case is that there is no statistical difference in the mean values of the two sets. As shown in Figure 4, the effect of expertise on time resulted in $F = 0.131$ and $p = 0.878$, so the test fails to reject the null hypothesis that there is no difference between the expertise groups with regard to time. The effect of expertise on accuracy resulted in $F = 0.863$ and $p = 0.429$, so this test also fails to reject the null hypothesis that there is no difference between the expertise groups with regard to accuracy. This shows that expertise has no statistical significance on the test results.

The MANOVA test considering the effect of the presence or absence of the FMPU report on time yielded $F = 18.40$ and $p = 0.000$. The presence or absence of the FMPU report effect on accuracy resulting in values $F = 18.99$ and $p = 0.000$. The Wilks Lambda statistic for the multivariate test itself yielded $F = 0.462$ and $p = 0.000$, which supports the validity of the test results. Thus, the null hypothesis that the means of the experimental and control groups are the same with regard to both time and accuracy is rejected, i.e., the means are significantly different for the experimental and control groups.

Based on the statistical evaluation, the metric that best describes the experimental results is the total experimental group mean compared with the control group mean. For additional analysis, the results of

Table 5. Final test results.

Trial Combination	Group	Mean Time (Minutes)	Standard Deviation	Mean Accuracy
All Correct	Control Group	40.70	19.10	100%
All Correct	Experimental	16.35	8.97	100%
Percent Decrease	60%			
All	Control Group	33.91	18.42	51%
All	Experimental	14.91	7.13	95%
Percent Decrease	53%			

only the groups who got everything correct are of interest as well. With the means between the experimental group and control group confirmed statistically as being different by the MANOVA test, the final results can now be considered. However as mentioned above, the five trial separation by expertise classification was shown to be not significant. Therefore, no *post hoc* analysis was conducted. The final test results are summarized in Table 5.

5. Conclusions

An important goal during the development of the Semi-Automated Crime-Specific Digital Triage Process Model and the Fast Modular Profiling Utility (FMPU) was to decrease the digital triage assessment time by at least 50% without any loss of accuracy compared with an *ad hoc* approach. The experimental results indicate that this goal has been met. Comparison of the mean time of all the subjects and the mean time of the subjects who obtained correct results reveals a decrease in the mean time of at least 50%. Meanwhile, no loss of accuracy was observed when the subjects used FMPU.

The level of expertise of the users was deemed to be not significant. Digital forensics expertise appears to have a slight effect, but Linux expertise may have more of an effect. The statistical analysis reveals that the presence or absence of FMPU has a statistically significant effect on time and accuracy. This leads to the conclusion that the Semi-Automated Digital Triage Process Model implemented with FMPU is useful in digital triage regardless of the expertise level of the user.

Our future research will focus on a more thorough evaluation of the digital triage model, including designing experiments that would examine additional variables, and increasing the numbers of subjects and subject groups in the qualitative and quantitative evaluations. We will also

focus on alternate implementations of the model and utility, additional user classifications and further crime template development.

References

[1] F. Adelstein, Live forensics: Diagnosing your system without killing it first, *Communications of the ACM*, vol. 49(2), pp. 63–66, 2005.

[2] V. Baryamureeba and F. Tushabe, The enhanced digital investigation process model, *Asian Journal of Information Technology*, vol. 5(7), pp. 790–794, 2006.

[3] N. Beebe and J. Clark, A hierarchical, objectives-based framework for the digital investigation process, *Digital Investigation*, vol. 2(2), pp. 147–167, 2005.

[4] A. Bogen and D. Dampier, Unifying computer forensics modeling approaches: A software engineering perspective, *Proceedings of the First International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 27–39, 2005.

[5] G. Cantrell, D. Dampier, Y. Dandass, N. Niu and A. Bogen, Research toward a partially-automated and crime-specific digital triage process model, *Computer and Information Science*, vol. 5(2), pp. 29–38, 2012.

[6] B. Carrier and E. Spafford, Getting physical with the digital investigation process, *International Journal of Digital Evidence*, vol. 2(2), 2003.

[7] B. Carrier and E. Spafford, An event-based digital forensic investigation framework, *Proceedings of the Digital Forensics Research Workshop*, 2004.

[8] H. Carvey, RegRipper (`regripper.wordpress.com`).

[9] H. Carvey, The Windows registry as a forensic resource, *Digital Investigation*, vol. 2(3), pp. 201–205, 2005.

[10] B. Dolan-Gavitt, Forensic analysis of the Windows registry in memory, *Digital Investigation*, vol. 5S, pp. S26–S32, 2008.

[11] D. Farmer and W. Venema, *Forensic Discovery*, Addison-Wesley, Upper Saddle River, New Jersey, 2004.

[12] A. Grillo, A. Lentini, G. Me and M. Ottoni, Fast user classifying to establish forensic analysis priorities, *Proceedings of the Fifth International Conference on IT Security Incident Management and IT Forensics*, pp. 69–77, 2009.

[13] T. Hill and P. Lewicki, *Statistics: Methods and Applications*, StatSoft, Tulsa, Oklahoma, 2006.

[14] K. Jones and R. Blani, Web Browser Forensics, Part 1, Symantec, Mountain View, California (www.symantec.com/connect/articles/web-browser-forensics-part-1), 2010.

[15] K. Jones and R. Blani, Web Browser Forensics, Part 2, Symantec, Mountain View, California (www.symantec.com/connect/articles/web-browser-forensics-part-2), 2010.

[16] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*, Addison-Wesley, Boston, Massachusetts, 2001.

[17] J. Oh, S. Lee, and S. Lee, Advanced evidence collection and analysis of web browser activity, *Digital Investigation*, vol. 8S, pp. S62–S70, 2011.

[18] M. Rogers, J. Goldman, R. Mislan, T. Wedge and S. Debrota, Computer forensics field triage process model, *Journal of Digital Forensics, Security and Law*, vol. 1(2), pp. 27–40, 2006.

III

FORENSIC TECHNIQUES

Chapter 7

REDUCING THE TIME REQUIRED FOR HASHING OPERATIONS

Frank Breitinger and Kaloyan Petrov

Abstract Due to the increasingly massive amounts of data that need to be analyzed in digital forensic investigations, it is necessary to automatically recognize suspect files and filter out non-relevant files. To achieve this goal, digital forensic practitioners employ hashing algorithms to classify files into known-good, known-bad and unknown files. However, a typical personal computer may store hundreds of thousands of files and the task becomes extremely time-consuming. This paper attempts to address the problem using a framework that speeds up processing by using multiple threads. Unlike a typical multithreading approach, where the hashing algorithm is performed by multiple threads, the proposed framework incorporates a dedicated prefetcher thread that reads files from a device. Experimental results demonstrate a runtime efficiency of nearly 40% over single threading.

Keywords: File hashing, runtime performance, file handling, prefetching

1. Introduction

The availability and use of electronic devices have increased massively. Traditional books, photos, letters and LPs have become ebooks, digital photos, email and music files. This transformation has also influenced the capacity of storage media, increasing from a few megabytes to terabytes. Thus, the amount of data gathered during digital forensic investigations has grown rapidly. To solve this data overload problem, it is necessary to employ automated techniques to distinguish relevant from non-relevant information.

To cope with the massive amount of data, digital forensic practitioners often use automated preprocessing that groups files into three categories: known-good, known-bad and unknown files. Operating system files and common application binaries are generally deemed to be known-good

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 101–117, 2013.

files and need not be inspected. The steps involved in classifying a file are to hash the file, compare the hash value against a hash database, and put the file into one of the three categories.

Because of the amount of data to be processed, runtime efficiency is an important issue. Thus, one property of hashing algorithms is the ease of computation, which is met by popular algorithms such as SHA-1 and MD5.

Meanwhile, the performance of modern hardware has increased considerably [11]. The addition of multiple cores and powerful GPUs may allow algorithm parallelization [1], but the principal bottleneck often involves loading data from files [19]. One approach for speeding up reading/writing is to use RAID systems that combine multiple disk drive components into a logical unit. Another approach is to use solid state drives (SSDs) that have higher throughputs than conventional hard disks [4]. However, because RAID and SSD technologies are not very widespread, a practical approach is to develop intelligent file handling solutions.

This paper presents a new framework for file handling during hashing. Instead of having several threads and/or cores that perform hashing independently, one thread is used for reading and the remaining threads are used for hashing. The framework can easily be used for other hashing algorithms or other tasks involving high data throughput.

2. Background

Hash functions have two basic properties, compression and ease of computation [10]. Hash functions are used in cryptography, databases [18] and for file identification [2]. In addition to traditional hash functions such as FNV [14], cryptographic hash functions such as SHA-1 [12] and MD5 [15], there are also similarity preserving hash methods such as `ssdeep` [9], `sdhash` [16] and `mrsh-v2` [7]. Similarity preserving hash algorithms are typically slower than traditional hash algorithms [7]. In order to optimize their runtime efficiency, most researchers have focused on algorithmic improvements [6, 8].

The most popular use case of cryptographic hash functions in digital forensics is known file detection. To detect files based on their hashes, it is necessary to rely on a database that includes a reference for each input file and its hash value. The most well-known database is the Reference Data Set (RDS) from the National Software Reference Library (NSRL) [13]. If the hash value of a file from a storage medium matches the hash value in the database, one can assume with some confidence that the file is the known file.

When a storage medium is examined, forensic software is used to hash the input, perform look-ups in the RDS and filter the non-relevant files. This reduces the amount of data that the investigator has to examine manually. Typically, the forensic software uses one thread to read and hash the file.

2.1 Similarity Preserving Hashing

Three similarity preserving hashing algorithms are used to demonstrate the effectiveness of the proposed framework:

- **ssdeep:** This algorithm [9], also called context triggered piecewise hashing, divides an input into approximately 64 pieces and hashes each piece separately. Instead of dividing the input into blocks of fixed length, it is divided based on the current context of seven bytes. The final hash value is the concatenation of all of the piecewise hashes where the context triggered piecewise hashing only uses the six least significant bits of each piecewise hash. This results in a Base64 sequence of approximately 64 characters.

- **sdhash:** This algorithm identifies "statistically-improbable features" using an entropy calculation [16]. The characteristic features, corresponding to a sequence of 64 bytes, are then hashed using SHA-1 and inserted into a Bloom filter [5]. Files are similar if they share identical features.

- **mrsh-v2:** This algorithm is based on `ssdeep`. The principal improvement is the removal of the restriction of 64 pieces, which creates a security issue [3]. Furthermore, instead of using a Base64 hash, `mrsh-v2` creates a sequence of Bloom filters [7].

2.2 Hash Function Run Time Efficiency

This section compares the runtime efficiency of the hashing algorithms included in the framework. All the tests used a 500 MiB file from `/dev/urandom` and the times were measured using the `time` command and the algorithm CPU-time (user time).

Table 1 presents the runtime efficiency comparison of the traditional cryptographic hash functions SHA-1 and MD5 versus the three similarity preserving hashing algorithms considered in this work. The results show that the traditional cryptographic hash functions outperform the similarity preserving hashing algorithms.

Table 1. Runtime efficiency comparison of hash functions.

	SHA-1	MD5	mrsh-v2	ssdeep 2.9	sdhash 2.0
Time	2.33 s	1.35 s	5.23 s	6.48 s	22.82 s
$\frac{\text{Algorithm}}{\text{SHA-1}}$	1.00	0.58	2.24	2.78	9.78

3. Parallelized File Hashing Framework

Our parallel framework for hashing (`pfh`) optimizes file handling during hashing. It is written in C++ and uses OpenMP 3.1 for multithreading; it is available at `www.dasec.h-da.de/staff/breitinger-frank`.

The framework is divided into two branches – simple multithreading (SMT) and multithreading with prefetching (MTP). SMT is used for comparison purposes and shows the benefits of the prefetcher.

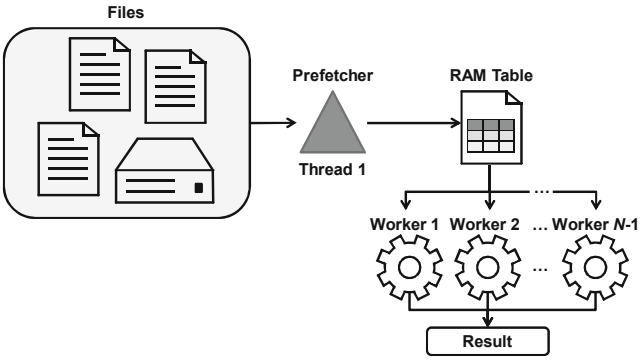


Figure 1. Operations of the framework.

Unlike traditional approaches where hash functions request a file and process it, the framework shown in Figure 1 includes a prefetching mechanism. The prefetcher handles file reading and is responsible for communications between the hard disk and RAM. The idea is that the critical resource bottleneck, the hard disk, should "work" all the time. Thus, the prefetcher produces an ongoing file request.

All the files are placed in RAM, which uses a RAM table to track the available storage. All remaining threads are "workers" and process the files from RAM using the hashing algorithms. After hashing the files, the outputs are denoted by result.

Depending on the computational efficiency of the hashing algorithm, there are two possibilities:

- If the hashing algorithm is fast, the worker threads are faster than the prefetching process and the workers must idle. However, the hard disk is at its limit and cannot process any faster.

- If the hashing algorithm is slow, the RAM table becomes full and cannot store any more files. This causes the prefetcher to idle. In this case, the prefetcher thread could turn into a normal worker and help process the files in the RAM table. Implementing this functionality is a part of our future work.

An alternative solution is to use a distributed system to further increase the performance. As will be demonstrated later, the limiting resource is the hard disk access time and not the computational power of the system. Thus, there is no improvement using a distributed system to hash inputs.

3.1 Command Line Parameters

Before describing the details of `pfh`, we introduce the command line options that allow rough configuration. Note that N denotes the number of processor cores in the system and the value of option `-p` is denoted by P where $P < N$.

- **c:** Mode of framework operation [optional].
- **d:** Directory to be hashed or file with digests [required].
- **r:** Recursive mode for directory traversal [optional].
- **p:** Number of prefetching threads [default is 1].
- **t:** Number of all threads [default is N].
- **h:** Hashing algorithm [default is `mrsh-v2`].
- **m:** Size of memory in MB [default is 20 MB].

A drawback is that all files larger than the RAM table are skipped. This problem will be corrected in a future version of the framework.

The framework operates in four modes:

- **HASH:** All the files are hashed using the specified algorithm and the results are printed to the standard output [default].
- **FULL:** The framework does an all-against-all comparison of all the files in `directory`.

- **<DIGEST>:** All the files in `directory` are hashed and compared against `DIGEST`, which is a single fingerprint.

  If parameter `-d` is a fingerprint file, then the framework compares `DIGEST` against all fingerprints in the file and skips hashing.

- **<FILENAME>:** `<FILENAME>` is replaced by a path to a file containing a list of valid hash values. The framework hashes all the files in `directory` and compares them against the list. This is similar to the `-m` option of `ssdeep`. If the signature is found in the list, then it is a valid match.

The following command executes the framework in the default mode using a RAM table of size 256 MB.

```
$ pfh -c hash -m 256 -d t5 -r
```

The `t5` directory is traversed recursively and all the hashes are sent to the standard output. If the `-t` and `-p` options are not specified, the program has $P = 1$ prefetching thread and $N - P$ hashing threads where N is the number of available processor cores.

3.2 Processing

Upon initialization, the framework creates the four building blocks: (i) options parsed; (ii) hashing interface created; (iii) RAM table created; and (iv) mode of operation. The directory containing the files is traversed and each file that passes the file size and access rights filter is added to the files-to-be-hashed-list.

The framework processing involves three stages: (i) reading/hashing files; (ii) comparing hash values; and (iii) presenting results whereby only the first and second stages are executed with multiple threads while the third stage is performed sequentially. Threads are created before the first stage and are finalized at the end of second stage. Thus, no time is lost for thread management (fork/join) during execution.

1. **Reading/Hashing Files:**

   - **SMT Branch:** Each of the N threads places its file in the RAM table and hashes it. All the threads continue until there are no more files in the queue.
     After being assigned to a role (reading or hashing), the threads enter a "work loop" for execution. Based on the return value, threads can change their role (e.g., if the RAM table is empty).
   - **MTP Branch:** Each thread receives a role assignment and begins execution (e.g., P prefetchers and $N - P$ hashing threads).

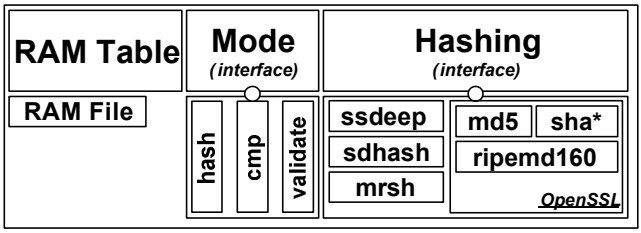


Figure 2. Objects in the framework.

2. **Comparing Hash Values:** This stage executes in parallel using the OpenMP "parallel for" clause, in which threads work on chunks of the global compare iterations.

   Scores from the comparison are held in an array, because if the threads print to screen, they have to synchronize and the speedup of parallelism is lost.

3. **Presenting Results:** The file path, hash value and the results are sent to the standard output.

3.3 Implementation Details

In addition to the two branches SMT/MTP and the operation modes, the framework comprises two objects, the RAM table and the hashing interface, which are shown in Figure 2.

SMT and MTP. Although SMT does not outperform MTP, we describe its implementation for completeness.

A configuration file called `configure.ac` is used to change the branch. This template is used by the configuration script when automake is executed. There are three options:

- **without-prefetching:** This option disables file prefetching and sets the branch to SMT (default: no, i.e., MTP mode).
- **with-timing:** This option enables timing (default: no). The supported times are total, compare, hashing, accumulated time for waiting for RAM and file, and reading from the disk. The option also provides throughput for hashing (MB/s) and comparisons (items/s).
- **with-stats:** This option enables statistics (default: no). Currently, only two state variables are added, waiting for a file and waiting for space in the RAM table.

RAM Table. The RAM table is the class responsible for holding files and synchronizing threads. Files are sent to the `ram_file` class, which provides functionality for reading files from the hard disk and processing them using the hash algorithm interface. `ram_table` uses two semaphores to implement the producer-consumer model. One semaphore is used to wait for free space in the RAM table and the other is used to wait for available/prefetched files in the RAM table.

The processing of the files in the table is based on two indices, fi and pi. The index fi denotes the number of files in the table and is set by the prefetcher; it increases by one after every insertion into the table. The index pi is for the worker threads; it increases by one every time a worker thread fetches a new file from the table. Thus, if $pi \geq fi$, threads have to wait for data.

To avoid race conditions, we use the OpenMP 3.1 "capture" clause. This enables a thread to take the current index and increase the global index in a single atomic operation. Thus, threads can work with RAM files without the need for locking or critical sections.

Interfaces. The framework accesses all hashing algorithms and modes via interfaces, enabling developers to add their own hashing algorithms. Realizations of the interfaces are written in their own `*.hpp` files and are included in the interface implementation file. Currently, the framework includes MD5, SHA1, SHA2, SHA3 and RIPEMD160 from the OpenSSL library and the `ssdeep`, `sdhash` and `mrsh-v2` similarity hashes.

The hashing interface `hash_alg.cpp` provides two extensions, one for hashing algorithms with character outputs and the other for byte outputs. The two extensions differ in the functions used to print and save a digest buffer. Member variables of the class are:

- **Output Type:** This could be a hex value or a string. For instance, MD5 yields a buffer holding a byte array, which has to be converted to a string.
- **Hash Digest Length:** This is required to print the hash value.
- **Minimum File Size:** This is required because some hashing algorithms have a minimum file size requirement (e.g., `ssdeep` requires 4,096 bytes).

Each hashing algorithm is implemented in its own file with the name `hash_alg_NAME.hpp`. Figures 3 and 4 show the changes needed to the `ssdeep` algorithm.

The `mode.h` interface allows the framework to operate in different ways after it is compiled. The interface itself consists of three virtual

```
01: class hash_al_ssdeep: public hash_alg_char_output{
02: public:
03:    int hash(uchar *in, uint inlen, uchar **out){
04:     *out = get_out();
05:     return (NULL == fuzzy_hash_buf_r((const uchar*)in, inlen, *out))
06:              ? -1: FUZZY_MAX_RESULT;
07:   };
08:
09:   int cmp(uchar *a, uchar *b, uint len){
10:     return fuzzy_compar_r(a,b);
11:   };
12:
13:   hash_alg_ssdeep(): hash_alg_char_output(){
14:     type            = HA_SSDEEP;
15:     max_result_size  =
16:     hash_digest_size = FUZZY_MAX_RESULT;
17:     min_file_size    = SSDEEP_MIN_FILE_SIZE;
18:   };
19: };
```

Figure 3. Framework extension for `ssdeep`.

```
01: if( 0 == htype.compare(0, 6, "ssdeep")){
02: h =  new hash_alg_ssdeep();
03: };
```

Figure 4. Initializing the hashing interface for `ssdeep`.

functions that represent the three steps of the framework: hashing files, comparing digests and printing results/digests.

Code Optimizations. Two code optimizations reduce the number of buffer allocations during execution. The first optimization pre-allocates all digest buffers; this reduces execution time because there are fewer calls of `new[]`. The second optimization reduces the memory footprint by grouping all the digest buffers into a single linear buffer. For instance, the GNU C library uses a header (two words for 8 bytes/32-bit and 16 bytes/64-bit systems) for each memory block. If a digest is allocated for MD5 (16 bytes), there is another 16 bytes (in 64-bit systems) of operating system administrative data (header).

Both optimizations are only available for hashing algorithms with a static hash value length. In the case of `mrsh-v2` and `sdhash`, which have a variable hash value length, the linear digest buffer cannot be allocated before hashing.

```
01: A(8), B(4), C(3), D(2), E(4), F(5) #Files order
02: TBL(0/10) #Table of size 10 with 0 space used
03: ------------------------------------
04: T1:PREF(A) -> TBL(8/10)
05: T1:PREF(B) -> TBL(8/10) -> WAIT(4) #Wait because only space of 2 is
                                        available
06: T2:HASH(A) -> TBL(0/10)
07: ------------------------------------
08: A(8), D(2), B(4), F(5), C(3), E(4) #Files order after balancing
```

Figure 5. RAM table balancing example.

3.4 Future Work

A future enhancement to the implementation will be the addition of a load balancing function. Load balancing would change the processing order of files, reducing both the waiting time for free table space and the fragmentation (empty space in table). A simple example is shown in Figure 5.

Table 2. Statistics of the t5 corpus.

	jpg	gif	doc	xls	ppt	html	pdf	txt
Amount	362	67	533	250	368	1,093	1,073	711

4. Experimental Results

The experimental evaluation of `pfh` used the t5 corpus [17], which contains 4,457 files of the types shown in Table 2. The unzipped size of files was 1.78 GB, corresponding to an average file size of 418.91 KB. The following tests are based on `ssdeep-2.9` and `sdhash-2.3`.

All the binaries were compiled using the same compiler and configuration options. The compiler flags included `-g0` to disable debugging, `-O2` to enable second level of optimization and `-march=native` to allow the use of CPU-specific instructions. The test environment was a server with the following components:

- **CPU:** Two Intel Xeon E5430 2.66 GHz × 4 cores
- **Hard Drive:** Seagate ES Series 250 GB (SATA 2) 8 MB Cache 7,200 RPM
- **RAM:** Eight 2 GB DDR2 FB-DIMM 667 MHz
- **Kernel:** Linux 2.6.32-279.11.1.el6.x86_64

Table 3. Runtime efficiency with `ssdeep` using standard output.

	Time	Difference	Terminal Command
Original	83.67 s	100.00%	`$ ssdeep -r t5`
SMT	69.05 s	82.52%	`$ pfh -d t5 -t 2 -h ssdeep`
MTP	52.19 s	62.37%	`$ pfh -d t5 -t 2 -h ssdeep`

- **GCC:** gcc-4.4.6-4.el6.x86_64

Three execution times were recorded:

- **Real Time:** This is the wall clock time from start to finish of the call. It corresponds to the elapsed time, including the time slices used by other processes and the time that the process spends in the blocked state.
- **User Time:** This is the amount of CPU time spent in user-mode code (outside the kernel) within the process. This coresponds to the actual CPU time used to execute the process. The time consumed by other processes and the time that the process spends in the blocked state do not count towards the user time.
- **Sys Time:** This is the amount of CPU time spent in the kernel within the process. This corresponds to the execution time spent in system calls within the kernel, as opposed to library code, which is still running in user space. Like the user time, this only includes the CPU time used by the process.

Since the framework improves the entirety of processing, only the real time is reported.

4.1 Runtime Efficiency

This section demonstrates the runtime improvement of the framework compared with the original implementation. The tests used the `-t 2` option, corresponding to one prefetching thread and one working thread.

Table 3 presents the results for `ssdeep`. Using SMT improves the basic hash algorithm runtime by approximately 17.5%. The MTP process shows an improvement of nearly 40%. The lower speedup of SMT is due to the lack of data in RAM. Having two threads means there are twice as many requests for file data, but with the same disk throughput. The threads are underfed and are forced to idle. In the case of MTP, it is a linear system – one reader and one hasher – which reduces the idle time for each thread.

Table 4. Runtime efficiency of the hash algorithms.

	MD5	SHA-1	mrsh-v2	ssdeep	sdhash
Original	51.65 s	52.35 s	75.61 s	83.67 s	145.38 s
MTP	51.74 s	51.64 s	51.79 s	52.19 s	89.09 s

Table 4 shows the runtime improvements for the hash algorithms. All the outputs were sent to `/dev/null` to eliminate any execution time deviation caused by printing. The main result is that prefetching has more impact on the similarity preserving hash algorithms, which are more computationally intensive than the cryptographic hash functions. MTP achieves similar runtimes for all the algorithms except `sdhash`. This shows that the limiting factor is the underlying hardware.

Table 5. Runtime efficiency with `FULL` mode [default `t` = 2].

	Time	Difference	Terminal Command
`ssdeep`	119.21 s	100.00%	`$ ssdeep -d -r t5`
MTP	68.34 s	57.33%	`$ pfh -h -c full -d t5 -t 8 -m 128`
`sdhash`	> 69 min	–	`$ sdhash -r -g -p 8 t5`
MTP	186.56 s	–	`$ pfh -h -c full -d t5 -t 8 -m 128`

Table 5 presents the results obtained using the `FULL` mode. This mode involves an all-against-all comparison in addition to hash value generation. In the case of `ssdeep` (rows 1 and 2), an improvement of nearly 45% was obtained. For `sdhash` (rows 3 and 4), the results are even better when the all-against-all comparison was stopped after 69 min and MTP only required 186 s.

4.2 Impact of Multiple Cores

This section explores the influence of multiple cores. The framework command line operation is:

```
$ pfh -h ALG -c hash -d t5 -m 256 -t XX > /dev/null
```

where `XX` is the number of cores/threads and `ALG` is the hash algorithm.

The test includes two runs, denoted by R1 and R2, shown in Tables 6 and 7. In R2, all of the files were pre-cached from R1.

R1 demonstrates that using multiple cores is important for slower algorithms like `sdhash`. For the faster traditional hashing algorithms, it does not scale well because the underlying hard disk is too slow and the prefetcher thread cannot fill the RAM table. R2 simulates fast hardware

Table 6. R1: Runtime efficiency with different numbers of threads.

		t = 2	**t = 4**	**t = 8**
`mrsh`	SMT	64.03 s	66.39 s	67.14 s
	MTP	51.79 s	51.81 s	52.02 s
`sdhash`	SMT	89.33 s	72.03 s	68.14 s
	MTP	89.09 s	51.90 s	52.08 s

Table 7. R2: Runtime efficiency with different numbers of threads and cached data.

		t = 2	**t = 4**	**t = 8**
`mrsh`	SMT	10.15 s	5.30 s	2.92 s
	MTP	17.68 s	6.23 s	2.90 s
`sdhash`	SMT	48.42 s	24.98 s	11.83 s
	MTP	88.15 s	31.12 s	15.07 s

because all the files are cached. As a result, the prefetcher thread is dispensable and SMT completes in less time.

Table 8. Runtime efficiency with different RAM table sizes.

	m=128	**m=256**	**m=51**
SMT	66.36s	66.39s	66.20s
MTP	51.62s	51.81s	51.72s

4.3 Impact of Memory Size

Table 8 shows that the size of the RAM table does not influence runtime efficiency. In general, there are two possibilities. The first is that there is a "slow" hashing algorithm and the prefetching thread is faster. Thus, the RAM table is always full because as soon as the worker thread fetches a file, the prefetcher adds the next one. The limiting source is thus the runtime efficiency of the algorithm.

The second possibility is that there is a "fast" hashing algorithm and the worker threads are faster. Thus, the RAM table is always empty because as soon as a new file is added, one worker processes the file. The limiting source is thus the underlying hardware.

Table 9. Impact of two prefetching threads.

Time	Difference	Terminal Command
52.05 s	100.00%	`$ pfh -t 8 -c hash -d t5 -h md5`
60.09 s	115.44%	`$ pfh -t 8 -c hash -d t5 -h md5 -p 2`

4.4 Impact of Multiple Prefetchers

Although the number of prefetcher threads is adjustable, tests showed that the default setting of one is the best choice. Table 9 verifies that having two prefetchers worsens the runtime by 15% due to the additional overhead.

Table 10. Summary of workstations used.

	Files	Size	Av. Size
Mac OSX	322,531	100.92 GB	328.08 KB
Windows 7	139,303	36.55 GB	275.13 KB

4.5 Impact on a Forensic Investigation

Next, we examine the improvement obtained using two workstations. The workstations listed in Table 10 were used. The results and projections are shown in Table 11.

Table 11. Investigation time using `ssdeep` on two workstations.

	Size	Standalone	SMT	MTP
Mac OSX	100.92 GB	99 min 51 s	73 min 43 s	56 min 43 s
Windows 7	36.55 GB	36 min 10 s	26 min 42 s	20 min 32 s

4.6 Parallelization Tool Comparison

This section compares `pfh` against Parallel and Parallel Processing Shell Script (`ppss`). Both Parallel and `ppss` execute commands, scripts or programs in parallel on local cores with multithreading and distribute the workload automatically to different threads.

Table 12 presents the results of using different parallelization tools. The main result is that MTP outperforms existing tools/scripts. Parallel and `ppss` both perform simple multithreading and should approximate

Table 12. Comparison of with different parallelization tools using `ssdeep`.

	Time	Difference	Terminal Command
Original	83.67 s	100.00%	`$ ssdeep -r t5 > /dev/null`
`ppss`	337.11 s	402.90%	`$ ppss -p 8 -d t5 -c 'ssdeep'`
Parallel	69.81 s	83.43%	`$ parallel ssdeep -- data/t5/*`
SMT	67.55 s	80.73%	`$ pfh -t 8 -c hash -d t5 -h ssdeep`
MTP	52.04 s	62.20%	`$ pfh -t 8 -c hash -d t5 -h ssdeep`

the SMT results. This holds true for Parallel, but in the case of `ppss`, the performance is worse due to an increase in I/O operations because `ppss` saves its state to the hard drive in the form of text files.

5. Conclusions

Current hashing techniques applied to entire filesystems are very time-consuming when implemented as single-threaded processes. The framework presented in this paper significantly speeds up hash value generation by including a separate prefetcher component. Experimental results demonstrate that an improvement of more than 40% is obtained for an all-against-all comparison compared with the standard `ssdeep` algorithm. In a real-world scenario, this results in a reduction in processing time from 1 hour and 39 minutes to 56 minutes without using any extra hardware.

The current implementation of the framework incorporates several cryptographic and similarity hash functions. Our future research will attempt to eliminate the limitation that only files smaller than the RAM table can be hashed.

Acknowledgements

This research was partially supported by the European Union Seventh Framework Programme (FP7/2007-2013) under Grant No. 257007. We also thank Nuno Brito with Serco Services and the European Space Agency (Darmstadt, Germany) for valuable ideas and discussions.

References

[1] D. Alcantara, A. Sharf, F. Abbasinejad, S. Sengupta, M. Mitzenmacher, J. Owens and N. Amenta, Real-time parallel hashing on the GPU, *ACM Transactions on Graphics*, vol. 28(5), article no. 154, 2009.

[2] C. Altheide and H. Carvey, *Digital Forensics with Open Source Tools*, Syngress, Waltham, Massachusetts, 2011.

[3] H. Baier and F. Breitinger, Security aspects of piecewise hashing in computer forensics, *Proceedings of the Sixth International Conference on IT Security Incident Management and IT Forensics*, pp. 21–36, 2011.

[4] A. Baxter, SSD vs. HDD (`www.storagereview.com/ssd_vs_hdd`), 2012.

[5] B. Bloom, Space/time trade-offs in hash coding with allowable errors, *Communications of the ACM*, vol. 13(7), pp. 422–426, 1970.

[6] F. Breitinger and H. Baier, Performance issues about context-triggered piecewise hashing, *Proceedings of the Third International ICST Conference on Digital Forensics and Cyber Crime*, pp. 141–155, 2011.

[7] F. Breitinger and H. Baier, Similarity preserving hashing: Eligible properties and a new algorithm `mrsh-v2`, *Proceedings of the Fourth International ICST Conference on Digital Forensics and Cyber Crime*, 2012.

[8] L. Chen and G. Wang, An efficient piecewise hashing method for computer forensics, *Proceedings of the First International Workshop on Knowledge Discovery and Data Mining*, pp. 635–638, 2008.

[9] J. Kornblum, Identifying almost identical files using context triggered piecewise hashing, *Digital Investigation*, vol. 3(S), pp. S91–S97, 2006.

[10] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Florida, 1997.

[11] G. Moore, Cramming more components onto integrated circuits, *Electronics Magazine*, pp. 114–117, April 19, 1965.

[12] National Institute of Standards and Technology, Secure Hash Standard, FIPS Publication 180-3, Gaithersburg, Maryland, 2008.

[13] National Institute of Standards and Technology, National Software Reference Library, Gaithersburg, Maryland (`www.nsrl.nist.gov`), 2012.

[14] L. Noll, FNV hash (`www.isthe.com/chongo/tech/comp/fnv/index.html`), 2012.

[15] R. Rivest, MD5 Message-Digest Algorithm, RFC 1321, 1992.

[16] V. Roussev, Data fingerprinting with similarity digests, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 207–226, 2010.

[17] V. Roussev, An evaluation of forensic similarity hashes, *Digital Investigation*, vol. 8(S), pp. S34–S41, 2011.

[18] S. Sumathi and S. Esakkirajan, *Fundamentals of Relational Database Management Systems*, Springer-Verlag, Berlin Heidelberg, Germany, 2010.

[19] S. Woerthmueller, Multithreaded file I/O, *Dr. Dobb's Journal*, September 28, 2009.

Chapter 8

HASH-BASED FILE CONTENT IDENTIFICATION USING DISTRIBUTED SYSTEMS

York Yannikos, Jonathan Schluessler, Martin Steinebach, Christian Winter and Kalman Graffi

Abstract A major challenge in digital forensics is the handling of very large amounts of data. Since forensic investigators often have to analyze several terabytes of data in a single case, efficient and effective tools for automatic data identification and filtering are required. A common data identification technique is to match the cryptographic hashes of files with hashes stored in blacklists and whitelists in order to identify contraband and harmless content, respectively. However, blacklists and whitelists are never complete and they miss most of the files encountered in investigations. Also, cryptographic hash matching fails when file content is altered even very slightly. This paper analyzes several distributed systems for their ability to support file content identification. A framework is presented for automated file content identification that searches for file hashes and collects, aggregates and presents the search results. Experiments demonstrate that the framework can provide identifying information for 26% of the test files from their hashed content, helping reduce the workload of forensic investigators.

Keywords: File content identification, hash values, P2P networks, search engines

1. Introduction

One of the principal challenges in digital forensics is the efficient and effective analysis of very large amounts of data. The most popular approach for filtering data in forensic investigations is to compute the cryptographic hashes of files and search for the hashes in large blacklists and whitelists. Since an important property of cryptographic hashes is collision resistance, they are ideally suited for file content identification. The

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 119–134, 2013.

cryptographic hash of a file is calculated solely from the file content – file metadata such as filename and timestamps are not considered. MD5 and SHA-1 are the most commonly used cryptographic hash functions that are used to compute file hashes.

Filtering files by searching for their hash values in blacklists and whitelists is effective and efficient. However, it is impossible to create lists that contain the file hashes of all harmful and harmless files. Moreover, the approach does not work well for multimedia files – changing the format of a picture, changes its file hash, and the new file hash does not match the original file hash.

This paper seeks an alternative to using blacklists and whitelists containing file hashes. It describes a framework that, given file hash values, can automatically engage sources such as P2P file sharing networks and web search engines to find useful information about the corresponding files. The useful information includes filenames because they typically describe the file contents. The framework aggregates, ranks and visually presents all the search results to assist forensic investigators in identifying file content without having to conduct manual reviews. Experimental results are presented to demonstrate the effectiveness and efficiency of the framework.

2. Hash-Based File Content Identification

Cryptographic hashes, especially MD5 and SHA-1, are commonly used in forensic investigations to identify content indexed in whitelists and blacklists. One of the largest publicly available databases is the NIST National Software Reference Library (NSRL), which reportedly contains about 26 million file hashes suitable for whitelisting [12]. Two publicly-available blacklisting databases are VirusTotal [19] and the Malware Hash Registry (MHR) [18] that can be searched for the hashes of malware files. Other databases maintained by law enforcement agencies contain the hashes of contraband files such as child pornography images, but these databases are generally not available to the public.

Tools have been developed to find known files in cloud system software and P2P client software. Examples are PeerLab [9] and FileMarshal [1] that find installed P2P clients and provide information about shared files and search history.

2.1 Advantages

Using cryptographic hashes for file content identification is effective because the hashes are designed to be resistant to pre-image and second pre-image attacks – it is extremely difficult to find the file corresponding

to a given hash value and to find another file that has the same hash value as a given file. Calculating file hashes using MD5 and SHA-1 is also efficient – the hash of a 100 MB file can be computed in less than one second on a standard computer system.

Another advantage is that only the file content is used to compute the file hash; no other metadata associated with the file is required. This means that a file that was modified to hide its illegal content, e.g., by giving it an innocuous filename or file extension or by changing its timestamps, can still be identified using its hash if the value is in a blacklist.

2.2 Limitations

Cryptographic hashing does not work well for multimedia files. For example, merely opening and saving a JPEG file results in a new file with a different hash value from the original file. Also, a contraband file that is already blacklisted may be distributed using anti-forensic techniques that make the automated identification of the file extremely difficult, if not impossible.

For example, a criminal entity could create a web server module as a content handler for contraband images. This module could be designed to randomly select and change a small number of pixels in each image before it is downloaded and viewed using a web browser. The modified image would be visually indistinguishable from the original contraband image. However, cryptographic hashing would fail to identify the modified image as being essentially the same as the original image.

Skin color detection algorithms are commonly used to detect child pornography. However, skin color detection is not a reliable technique because it has a high false positive rate. Also, skin color detection alone would not be able to discriminate between child pornography and legal pornography.

Steinebach, *et al.* [16] have proposed an efficient and accurate approach that uses robust hashing to identify blacklisted child pornography. However, robust hashing has not as yet been used for file content indexing by P2P protocols, web search engines and most hash databases. For this reason, we do not consider robust hashing in this paper.

3. Distributed Systems for Hash-Based Searches

This section examines various distributed systems to assess their suitability for file hash searches. In particular, the section focuses on P2P file sharing networks, web search engines and online databases of indexed file hashes.

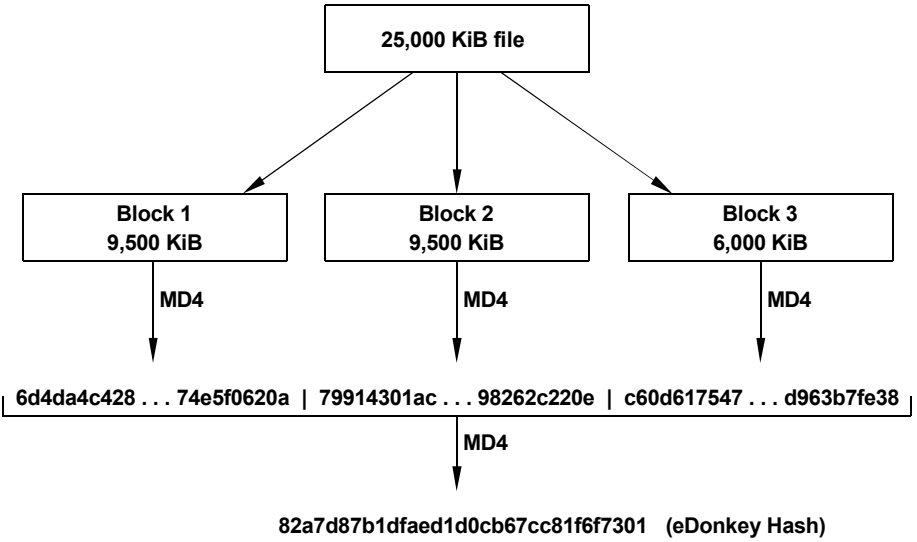


Figure 1. Creation of an eDonkey hash of a sample 25,000 KiB file.

3.1 P2P File Sharing Networks

P2P file sharing networks are immensely popular. A recent Cisco study [3] reported that P2P networks contributed to about 15% of the global IP traffic in 2011. Another study [15] identified P2P traffic as the largest share of Internet traffic in every region of the world, ranging from 42.5% in Northern Africa to almost 70% in Eastern Europe. The most popular P2P protocols are BitTorrent, eDonkey and Gnutella.

Although BitTorrent [4] is the most popular P2P protocol with 150 million active users per month [2], the protocol does not allow hash-based file searches. BitTorrent uses "BitTorrent info hashes" (BTIHs) instead of cryptographic hashes of the files that are shared. Since BTIHs are created from data other than just file content (e.g., filenames), BitTorrent is not suitable for our hash-based file search approach.

Another popular P2P protocol is eDonkey with 1.2 million concurrent online users as of August 2012 (aggregated from [6]). The eDonkey protocol is well suited to hash-based file searches because the "eDonkey hash" used for indexing is a specific application of the MD4 cryptographic hash [10]. Figure 1 illustrates the process of creating an eDonkey hash. A file is divided into equally-sized blocks of 9,500 KiB and an MD4 hash value is computed for each block. The resulting hashes are then concatenated to a single byte sequence for which the final MD4 hash (eDonkey hash) is computed.

An eDonkey peer typically sends its hash search request to one of several eDonkey servers, which replies with a list of search results. eDonkey also allows decentralized searches using its kad network, a specific implementation of the Kademlia distributed hash table [11]. Interested readers are referred to [17] for additional details about kad.

The Gnutella protocol uses SHA-1 for indexing shared files. Searching for SHA-1 hashes was initially supported by Gnutella, but it is now disabled in most Gnutella clients for performance reasons. The `gtk-gnutella` client still supports SHA-1 hash searches. However, this client yields very few results because a SHA-1 hash search request is dropped by all other clients that do not support SHA-1 hash searches. Also, in test runs, we have observed that `gtk-gnutella` has very long response times for SHA-1 hash searches. Therefore, Gnutella is not suitable for hash-based file searches for reasons of effectiveness and efficiency.

3.2 Web Search Engines

According to [13], Google is by far the most popular web search engine with about 84.4% market share; next come Yahoo! with 7.6% and Bing with 4.3%. Since the success of search engines is based on gathering as much Internet content as possible while applying powerful indexing mechanisms to support fast querying of very large data sets, they provide an excellent basis for hash-based file searches. Because many web sites (e.g., file hosting services) also provide file hashes such as MD5 or SHA-1 as checksums, it is very likely to find valuable information such as a popular filename when searching for a file hash. Figure 2 shows a screenshot of a file hosting website [5] that provides useful information about a file with a specific MD5 file hash value.

3.3 Online Hash Databases

In addition to NIST's National Software Reference Library (NSRL) [12], several other hash databases can be searched online or downloaded and searched locally. One example is the Malware Hash Registry (MHR) [18], which contains the hashes of files identified as malware. Another is SANS Internet Storm Center's Hash Database (ISC) [14], which reportedly searches the NSRL as well as MHR for hashes. MHR and ISC both support search queries via the DNS protocol by creating DNS TXT requests that use specific file hashes as a subdomain of a given DNS zone.

Figure 3 shows an example of querying the ISC database using the DNS protocol. The source "NIST" specifies the NSRL database. The result contains the indexed filename `Linux_DR.066` belonging to the hash.

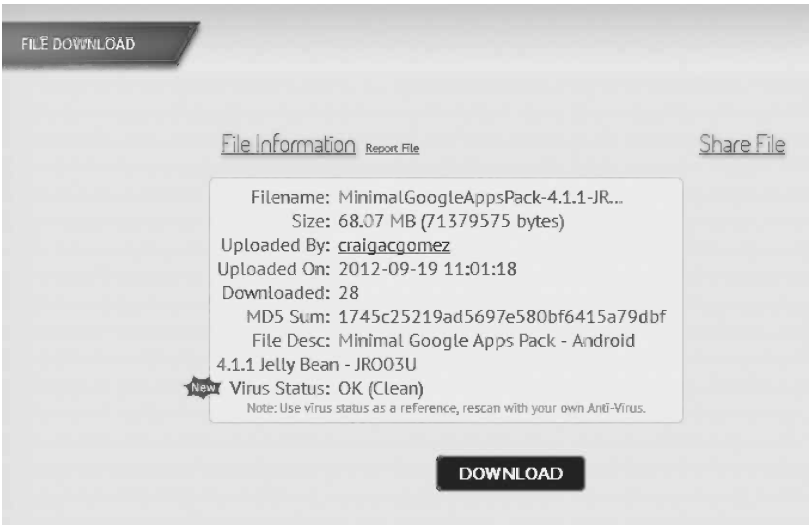


Figure 2. File hosting website with useful information about an MD5 file hash.

```
> dig +short B47139415F735A98069ACE824A114399.md5.dshield.org
              TXT "Linux_DR.066 | NIST"
```

Figure 3. Querying the ISC database for a file hash using `dig`.

4. File Content Identification Framework

When searching for a specific file hash, the most important information directly connected with the hash is a commonly-used filename because it typically describes the file content in a succinct manner. In order to automatically search different sources and collect the results, we developed a prototype that implements the following modules to search P2P file sharing networks, web search engines and hash databases for file hashes:

- **eDonkey Module:** This module is used to search eDonkey P2P networks for eDonkey hashes.
- **Google Module:** This module is used to search Google for MD5 and SHA-1 hashes.
- **Yahoo! Module:** This module is used to search Yahoo! for MD5 and SHA-1 hashes.

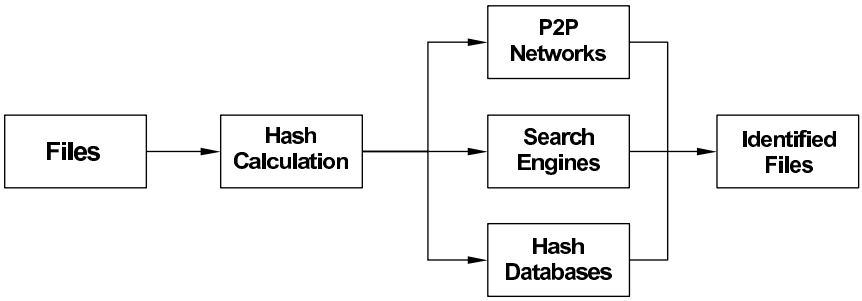


Figure 4. File content identification using cryptographic hashes.

- **NSRL Module:** This module is used to search the NSRL for MD5 hashes. A copy of the NSRL was downloaded in advance to perform local database searches.
- **ISC Module:** This module is used to search ISC for MD5 hashes.
- **MHR Module:** This module is used to search MHR for MD5 hashes.

Figure 4 shows the file content identification process of our framework. The process has five steps:

1. Select a directory containing the input files.
2. Calculate the MD5, SHA-1 and eDonkey hashes for each file in the directory and its subdirectories.
3. Use the hashes to query P2P networks, search engines and hash databases.
4. Collect all the results and aggregate the identical results.
5. Present the results in a ranked list.

5. Evaluation

In order to evaluate our framework, we used a test set of 1,473 different files with a total size of about 13 GB. Although the test set was rather small, it provided a good representation of files that would be of interest if encountered during a forensic investigation. The test set included the following categories of files:

- **Music Files:** A total of 650 audio files, mainly containing music. These files were randomly selected from three large music libraries, several music download websites and sample file sets.

- **Picture Files:** More than 500 pictures, mainly from several public and private picture collections (e.g., 4chan image board).
- **Video Files:** A total of 34 video files, mainly YouTube videos, entire movies and episodes of popular TV series.
- **Document Files:** A total of 240 documents, including e-books in the PDF, EPUB and DOC formats, mainly from a private repository.
- **Miscellaneous Files:** Several archive files, Windows and Linux executables, malware and other miscellaneous files, mainly from a private download repository.

5.1 Search Result Characteristics

Our framework collects metadata found by searching P2P file sharing networks, search engines and hash databases. The search results are presented in a ranked form based on how many different sources found the same metadata. In general, the more specific a filename that is found when searching for a file hash, the higher the filename is ranked. Along with the filename, the framework also collects other metadata (e.g., URLs and timestamps), if available.

Table 1 shows the results of a hash-based search using the framework to identify file content. Specifically, the table shows the filenames found for three sample files from the test set: one audio file, one picture file and one video file. As mentioned above, locally-available metadata such as a local filename is not considered by our framework because it cannot be trusted and may be forged to hide incriminating information. The sample video file turned out to be a good example of the usefulness of the framework. Since the local filename is `Pulp Fiction 1994.avi`, a forensic investigator is likely to conclude that the file contains the popular movie *Pulp Fiction* directed by Quentin Tarantino. However, in eDonkey networks, our framework found several filenames describing files with pornographic content. Most of the filenames found in our evaluation provided enough information to identify the content of the corresponding files.

5.2 Effectiveness

For 382 files (26%) in the test set of 1,473 files, we were able to find a commonly-used filename that helped identify the file content. 220 of these files (almost 58%) yielded unique results, i.e., only a single source produced results for these files. Figure 5 shows the total percentage of

Table 1. Results of a hash-based search.

Sample File	**Source**	**#Results**	**Most Common Filenames (sorted by #results in descending order)**
Picture File	Google	1,100	`Lighthouse.jpg` `8969288f4245120e7c3870287cce0ff3.jpg`
	Yahoo!	7	`Lighthouse.jpg`
	eDonkey	7	`Lighthouse.jpg` `sfondo hd (3).jpg`
	ISC	1	`Lighthouse.jpg`

Audio File	eDonkey	31	`Madonna feat. Nicki Minaj M.I.A. Give Me All Your Luvin.mp3`
			`Madonna - Give me all your lovin [ft.Nicki Minaj & M.I.A.].mp3`
			`Madonna ft. Nicki Minaj & M.I.A. - Give Me All Your Luvin'.mp3`

Video File	eDonkey	130	`La.Loca.De.la.Luna.Llena.Spanish. XXX.DVDRip.www.365(...).mpg`
			`La Loca De La Luna Llena (Cine Porno Xxx Anastasia Mayo(...).avi`
			`MARRANILLAS TORBE...avi`
			`Pelis porno - La Loca De La Luna Llena (Cine Porno Xxx(...).mpg`

files for which the individual sources successfully found a filename, along with the aggregated numbers.

Most of the results were obtained by searching eDonkey and Google. In comparison, searching Yahoo!, ISC and MHR produced no additional results. Apart from Google and eDonkey, the NSRL was the only other source that produced unique results.

Figure 6 shows the search results for each category. eDonkey was very effective at finding filenames using the hashes of video files – it produced results for 53% of the video files in the test set. Furthermore, eDonkey found filenames for the hashes of 18% of the picture files, 19% of the audio files and 24% of the miscellaneous files.

Google yielded the most results for miscellaneous files; it found filenames for 66% of the hashes. The NSRL and ISC hash databases yielded the filenames of most system files, sample wallpapers and sample videos that are typically whitelisted. Searching for documents such as e-books and Microsoft Word files turned out to be ineffective. Google was still

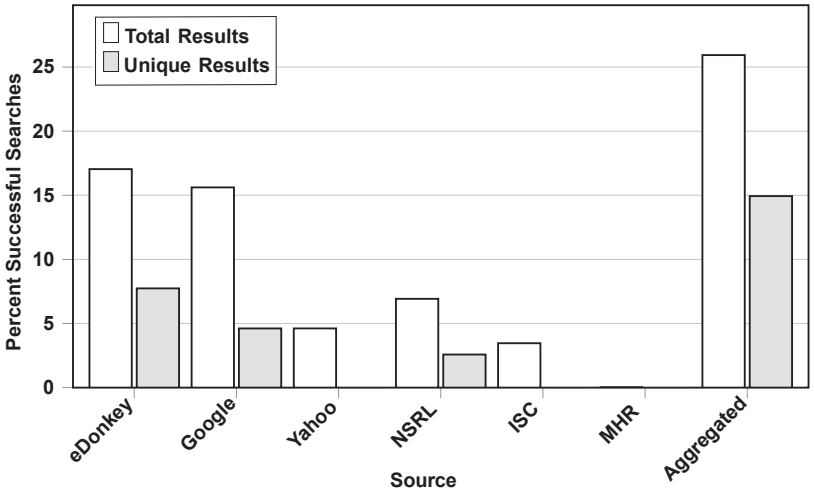


Figure 5. Successful searches for each source and aggregated numbers.

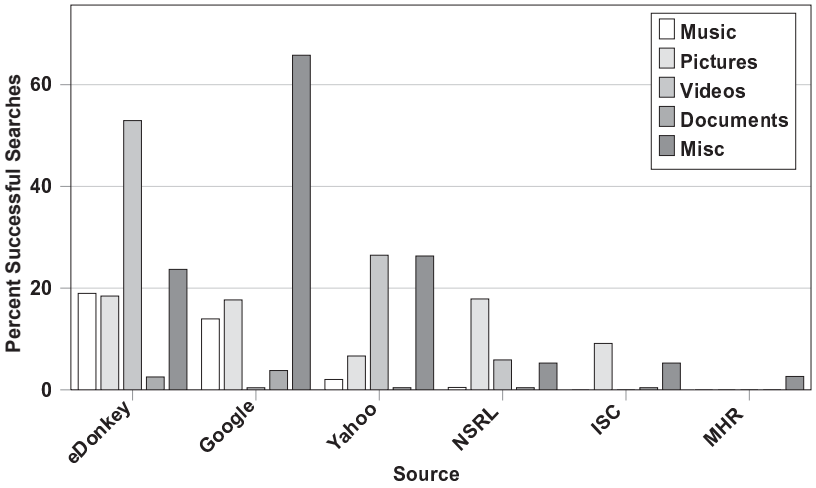


Figure 6. Successful searches for each source by category.

the most successful source, but it found filenames for just 3.8% of the document file hashes.

In addition to MD5 and SHA-1 hashes, we used Google to search for hashes in the SHA-2 family because they are supported by many forensic analysis tools. Figure 7 shows the results. The SHA-224, SHA-256, SHA-384 and SHA-256 hashes only yielded a small number of results with no unique results. This leads us to believe that using MD5 and SHA-1 is adequate for reliable file content identification based on cryptographic hashes.

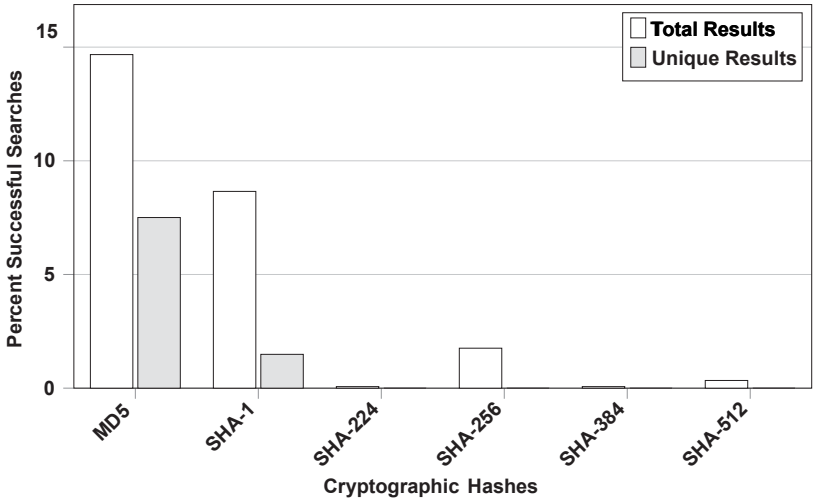


Figure 7. Successful Google searches using cryptographic hashes.

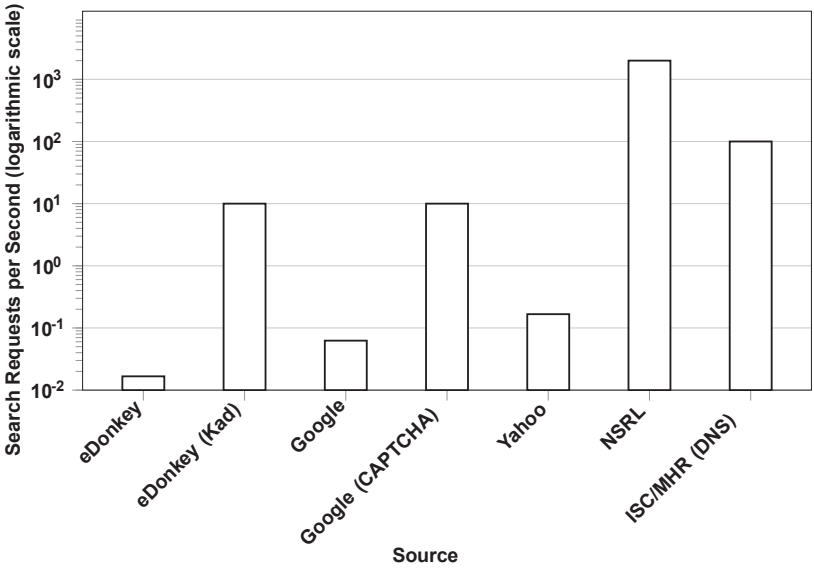


Figure 8. Search requests per second for each source.

5.3 Efficiency

To evaluate the efficiency of our framework, we measured the average number of search requests per second that were sent to each source. Figure 8 shows the results. We found that some sources (e.g., hash databases) could be searched very quickly. Other sources implement

mechanisms such as CAPTCHAs to throttle the number of requests received during a given time period.

eDonkey servers, which are used for file search, generally have security mechanisms that prevent request flooding. We discovered that even if just one request was sent every 30 seconds, the servers quickly stopped replying to additional requests. Eventually, we found that sending one request every 60 seconds produced results without being blocked. However, using the decentralized kad network of eDonkey (where peers are directly contacted with search requests), we were able to send requests much faster, at a rate of about 10 requests per second.

The locally-available NSRL hash database was the fastest; it handled about 2,000 requests per second. The MHR and ISC databases were also fast, about 100 search requests per second were possible using their DNS interfaces.

In the case of Google and Yahoo!, the average rates for search requests were one request in sixteen seconds and one request in six seconds, respectively. Sending faster requests caused Yahoo! to completely ban our IP addresses, with increasing ban durations of up to 24 hours. On the other hand, sending faster requests to Google required the solution of CAPTCHAs.

For Google, we observed that the search request rate affected the number of search requests allowed during a specific time period before a CAPTCHA was shown. When requests were sent more often than one every sixteen seconds, Google showed a CAPTCHA after about 180 requests. When the request rate was increased to one request every 0.2 to two seconds, Google showed a CAPTCHA after about 75 requests. However, when requests were sent faster than one every 0.2 seconds, Google surprisingly allowed up to 1,000 search requests until a CAPTCHA was shown. By adding an automatic CAPTCHA-solving service, which typically requires less than 20 seconds per CAPTCHA, we could theoretically increase the throughput of Google searches to about ten requests per second. However, Google eventually started to ban our IP addresses, so we did not examine this issue any further.

Figure 9 shows the number of Google search requests (with error bars) before a CAPTCHA was shown in relation to the duration between search requests.

6. Use Case Scenario

A good use case scenario for our framework is a forensic investigation where seized storage devices have to be analyzed. After creating forensically-sound images of the seized devices, an investigator

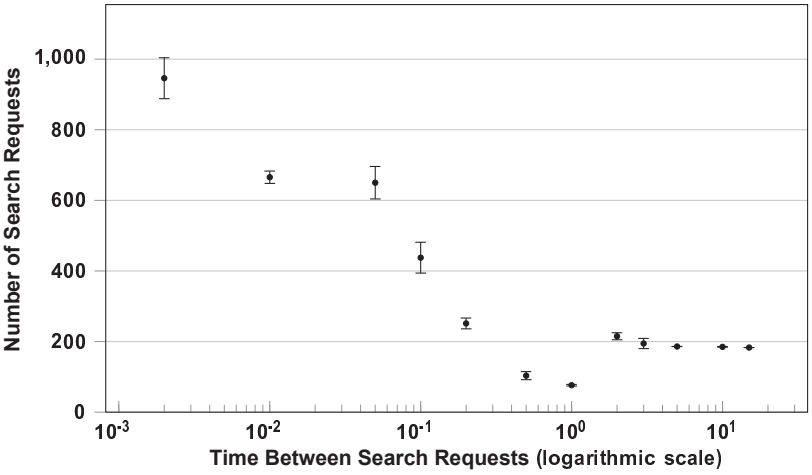


Figure 9. Google search requests.

would typically use a standard forensic tool to perform filesystem analysis. Next, data recovery would target the unallocated space to find previously-deleted files that have not been overwritten. The original filenames of recovered files are often missing, especially when a file carver has been used (e.g., because filesystem information may no longer be available). In the worst (but not atypical) case, the forensic investigator would end up with a large amount of files with non-descriptive filenames and almost no information about their content.

The recovered data must then be reviewed in order to find incriminating and exonerating material. Filtering techniques using whitelists and blacklists of cryptographic hashes would then be used to reduce the amount of data to be reviewed. This is where our framework can significantly support the forensic investigator – instead of only filtering based on incomplete hash databases, the investigator can employ our framework and feed all the data to be reviewed into it.

Our framework would automatically start searching multiple sources for information about the data. Whitelisted files such as system files and common application files would be automatically filtered. Blacklisted files would be marked as such and would be presented to the investigator together with aggregated and ranked information about files that are neither whitelisted nor blacklisted, but for which information such as a common filename has been found by searching P2P file sharing networks or using search engines.

By going through the ranked list of search results that contain, for example, content-describing filenames, a forensic investigator can find valuable information about the files of interest and their content without

having to do a manual review. The investigator can then identify the files that should be eliminated due to irrelevant content and the files to be reviewed first because the results indicate that they have incriminating material. Searching for file hashes in distributed systems with large amounts of dynamically-changing content can greatly complement static filtering techniques based on whitelists and blacklists.

7. Conclusions

Distributed systems such as P2P networks and search engines can be used to gather information about files based on their file hashes. The framework created to automate file content identification is very effective at searching these distributed systems and aggregating, ranking and presenting the search results. This could greatly assist forensic investigators in gaining knowledge about file contents without conducting manual reviews.

The experimental results demonstrate that searching eDonkey networks is very effective for multimedia data such as video files, audio files and pictures. Google and the NSRL are also well suited to hash-based file identification. However, Yahoo! and the ISC and MHR hash databases did not produce any unique search results, and may be eliminated from consideration in favor of eDonkey, Google and the NSRL.

The hash databases were the fastest to search, followed by eDonkey's kad network and then Google (albeit with an automatic CAPTCHA solver). The experiments also demonstrated that MD5 and SHA-1 are still the most commonly used file hashes and are, therefore, well suited to hash-based file searches. Other hashes, such as SHA-2, are rarely used and may be eliminated from consideration at this time.

Our future research will identify other distributed systems that are suitable for hash-based file searches (e.g., P2P file sharing networks like DirectConnect). Also, databases providing specific multimedia content, such as Flickr [20], Grooveshark [7] and IMDb [8], will also be investigated. Future improvements to the framework include creating a client-server model for parallelizing search tasks and leveraging a cloud service infrastructure for faster distributed file hash searches.

Acknowledgement

This research was supported by the Center for Advanced Security Research Darmstadt (CASED), Darmstadt, Germany.

References

[1] F. Adelstein and R. Joyce, File Marshal: Automatic extraction of peer-to-peer data, *Digital Investigation*, vol. 4(S), pp. S43–S48, 2007.

[2] BitTorrent, BitTorrent and μTorrent software surpass 150 million user milestone; announce new consumer electronics partnerships, Press Release, San Francisco, California (`www.bittorrent.com/intl/es/company/about/ces_2012_150m_users`), January 9, 2012.

[3] Cisco Systems, Cisco Visual Networking Index: Forecast and Methodology, White Paper, San Jose, California (`www.cisco .com/en/US/solutions/collateral/ns341/ ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`), 2012.

[4] B. Cohen, Incentives build robustness in BitTorrent, *Proceedings of the First International Workshop on the Economics of Peer-to-Peer Systems*, 2003.

[5] Dev-Host, The ultimate free file hosting/file sharing service, Los Angeles, California (`d-h.st`).

[6] eMule-MODs.de, Server List for eDonkey and eMule (`www.emule-mods.de/?servermet=show`).

[7] Escape Media Group, Grooveshark, Gainesville, Florida (`www.grooveshark.com`).

[8] IMDb.com, Internet Movie Database, Seattle, Washington (`www.imdb.com`).

[9] Kuiper Forensics, PeerLab – Scanning and evaluation of P2P applications, Mainz, Germany (`www.kuiper.de/index.php/en/peerlab`).

[10] Y. Kulbak and D. Bickson, The eMule Protocol Specification, Technical Report, School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel, 2005.

[11] P. Maymounkov and D. Mazieres, Kademlia: A peer-to-peer information system based on the XOR metric, *Proceedings of the First International Workshop on Peer-to-Peer Systems*, pp. 53–65, 2002.

[12] National Institute of Standards and Technology, National Software Reference Library, Gaithersburg, Maryland (`www.nsrl.nist.gov`).

[13] Net Applications, Desktop Search Engine Market Share (`www.netmarketshare.com/ search-engine-market-share.aspx? qprd=4&qpcustomd=0`), October 2012.

[14] SANS Internet Storm Center, Hash Database, SANS Institute, Bethesda, Maryland (isc.sans.edu/tools/hashsearch.html).

[15] H. Schulze and K. Mochalski, Internet Study 2008/2009, ipoque, Leipzig, Germany (www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf), 2009.

[16] M. Steinebach, H. Liu and Y. Yannikos, Forbild: Efficient robust image hashing, *Proceedings of the SPIE Conference on Media Watermarking, Security and Forensics*, vol. 8303, 2012.

[17] M. Steiner, T. En-Najjary and E. Biersack, A global view of kad, *Proceedings of the Seventh ACM SIGCOMM Conference on Internet Measurement*, pp. 117–122, 2007.

[18] Team Cymru, Malware Hash Registry (MHR), Lake Mary, Florida (www.team-cymru.org/Services/MHR).

[19] VirusTotal Team, VirusTotal, Malaga, Spain (www.virustotal.com).

[20] Yahoo! Flickr, Sunnyvale, California (www.flickr.com).

Chapter 9

CREATING SUPER TIMELINES IN WINDOWS INVESTIGATIONS

Stephen Esposito and Gilbert Peterson

Abstract As the applications and adoption of networked electronic devices grow, their use in conjunction with crimes also increases. Extracting probative evidence from these devices requires experienced digital forensic practitioners to use specialized tools that help interpret the raw binary data present in digital media. After the evidentiary artifacts are collected, an important goal of the practitioner is to assemble a narrative that describes when the events of interest occurred based on the timestamps of the artifacts. Unfortunately, generating and evaluating super timelines is a manual and labor-intensive process. This paper describes a technique that aids the practitioner in this process by generating queries that extract and connect the temporal artifacts, and produce concise timelines. Application of the queries to a simulated incident demonstrates their ability to reduce the number of artifacts from hundreds of thousands artifacts to a few hundred or less, and to facilitate the understanding of the activities surrounding the incident.

Keywords: Windows forensics, events, artifacts, super timelines

1. Introduction

Computers and digital devices are ubiquitous, as a result these devices are often present at crime scenes or are used to commit crimes. This makes the devices valuable sources of evidence in criminal investigations. One of the key tasks when examining a device is to understand how the temporal nature of the relevant data items correlate with each other. For example, while it is important to identify when a virus first appeared in a network, it is also important to understand how other temporal data items correlate with the arrival of the virus, such as the users that were logged in and their activities at the time of the infection. Connecting these temporal items together creates a timeline of events

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 135–144, 2013.

that can explain and confirm the details of an incident to include who, what, where and when.

This paper discusses how a timeline of events can be developed and processed into a concise listing, and subsequently analyzed to correlate temporal items and produce knowledge about how an incident occurred. Four queries are presented that help convert raw file and event log timestamps to a general timeline of what the user and executing processes were doing on a Windows system during the period of interest. The results of testing the queries on a simulated incident demonstrate a significant reduction in the number of data items to be evaluated, which greatly reduces the evidence analysis and correlation time.

2. Related Work

Creating a timeline of the various events that occurred during an incident is one of the key tasks performed by a digital forensic practitioner [4]. Several tools have been developed for extracting temporal artifacts from specific data sources. For example, `fls` [1] extracts filesystem metadata while RegRipper [2] and `yaru` [9] extract Windows registry details that can identify the ordering of events. Pasco [5] and Web Historian [6] are web history extraction tools that parse Internet histories. EnCase and FTK also provide search functionality to find timestamp information, but neither tool has a super timeline capability [7]. The two tools produce a narrow view of user activities, requiring the practitioner to manually standardize, connect and correlate the data to produce a quality super timeline.

Instead of running each tool individually, the `log2timeline` tool can be used to extract all the artifacts and more in order to produce a single super timeline for analysis. The tool engages a Perl script to parse a file structure and artifacts. The tool has several input and output modules that enable it to be used in a variety of operating system environments. The practitioner identifies the operating system when executing `log2timeline` and the tool automatically uses all the modules available for the operating system.

The SIMILE output module [8] is a key `log2timeline` module. The module provides output in a standardized format that enables a SIMILE widget to read and display a moving timeline of information. While this may seem like an ideal way to view `log2timeline` output, it is difficult to use the tool to view more than 200 items because the SIMILE widget loads and operates very slowly. Because of this limitation, a better option is to produce the output as a comma separated value (CSV) file. This format simplifies the importation of data to a database.

3. Timestamp Anti-Forensics

An important issue to discuss at this stage is the possibility of anti-forensic tools being used to alter timestamp data. For example, the Timestomp [3] tool can alter the modification, access and creation (MAC) timestamps in the $STANDARD_INFORMATION attribute of a master file table (MFT) entry, albeit not the four timestamps present in the $FILE_NAME attribute. An attacker intending to cover his tracks can use Timestomp to alter the timestamps of incriminating files. This makes the creation of an accurate timeline much more difficult.

If tampering is suspected, all eight timestamps in the MFT, the file metadata timestamps and the recently-used registry keys should be correlated to identify tampering. It is also important to search for traces of the execution of tools such as Timestomp and any artifacts that may reside in the filesystem.

4. Timeline Artifact Analysis

Temporal data analysis comprises the collection and extraction of artifacts, correlation of the artifacts, and the determination of when an activity occurred. Our collection and extraction process involves generating an image of the hard drive and executing the `log2timeline` tool on the image to extract timestamp data.

The `log2timeline` CSV output file was imported into a Microsoft Access database with the fields shown in Table 1. SQL queries were developed to produce timelines of the temporal artifacts that correlate the timestamp data, which were subsequently used to reconstruct the timeline of events that occurred just before, during and after the incident. SQL queries providing the following information were developed to aid a digital forensic practitioner in correlating artifacts:

- All user activities (items not associated with a user are excluded).
- Web history traffic with user login and logoff times.
  - Refined query with keyword search.
  - Additional refined query to add a user.
- Documents recently accessed by a user.
- All login, logoff and program execution times within a specific timeframe.

The SQL queries were designed for reusability and to reduce the time and effort required to investigate potentially hundreds of thousands of

Table 1. Data items in the `log2timeline` CSV file.

Field Name	Data Type
ID	AutoNumber
date	Date/Time
time	Date/Time
timezone	Text
MACB	Text
source	Text
sourcetype	Text
type	Text
user	Text
host	Text
short	Memo
file_description	Memo
version	Number
filename	Memo
inode	Number
notes	Memo
format	Text
extra	Text

artifacts. The keywords and time frames may change, but the queries would remain the same.

4.1 User Program Activity Query

If no information is available about an incident, then a listing of the users and what each user did on the system is a good way to begin the search for information. The user program activity query gives the practitioner an overall view of a user's general activity. The query displays artifacts like the user assist registry key, which is created when the user executes a program such as Notepad. But it does not display operating system artifacts such as MFT and prefetch artifacts for the same execution of Notepad. If there is no user associated with a timestamp, then `log2timeline` places a "–" in the user field, which the query uses to identify non-user events.

A user program activity query has the following format:

```
SELECT L2t.[date], L2t.[time], L2t.[source],
       L2t.[sourcetype], L2t.[type], L2t.[user],
       L2t.[short], L2t.[desc], L2t.[filename],
       L2t.[notes], L2t.[format], L2t.[extra]
FROM L2t
WHERE ((NOT (L2t.[user])="-"))
ORDER BY L2t.[date] DESC, L2t.[time] DESC;
```

4.2 User Web History Query

Digital forensic practitioners are often required to investigate incidents involving the intentional or unintentional access to an inappropriate or unauthorized website. In such a situation, a practitioner would be interested in extracting web history artifacts to identify the websites accessed by a user. The login and logoff events corresponding to the user can be found by querying for the audit log event IDs 528 and 538 in a Windows XP system, and the event IDs 4624 and 4647 in a Windows 7 system. Web history artifacts may be found by searching for artifacts for which the source is logged as `WEBHIST`. The following query uses the term `(filename) Like "*Security/528;*"` to search for `Security/528;` anywhere in an artifact.

```
SELECT date, time, source, user, filename
FROM log2timeline
WHERE (source = "WEBHIST") OR ((filename) Like "*Security/528;*")
                           OR ((filename) Like "*Security/538;*")
ORDER BY date DESC, time DESC;
```

This is different from only using `source = "WEBHIST"` because the resulting query will only find artifacts where the source is logged exactly as `WEBHIST`.

4.3 Recent Document Access Query

Other important information relates to documents "touched" by a user. In a Windows system, whenever a file or resource is touched by a user, a link file is created in the user's recently accessed folder. The following query for all recent documents can provide useful information about user activities in an investigation:

```
SELECT L2t.date, L2t.time, L2t.source, L2t.sourcetype,
       L2t.type, L2t.user, L2t.filename, L2t.short,
       L2t.desc, L2t.notes, L2t.extra
FROM L2t
WHERE (((L2t.desc) Like "*recent*"))
ORDER BY L2t.date DESC, L2t.time DESC;
```

4.4 Processes Executed During User Login Timeframe Query

After a user or timeframe has been identified, a query for artifacts of executable programs during a user's login timeframe allows a practitioner to focus on the main activity on the computer. Once the practitioner understands the user's activities, the query can be expanded to

include all the artifacts during the timeframe in order to add correlative temporal artifacts to the timeline. The basic query for executable artifacts in a specific timeframe is:

```
SELECT date, time, source, user, file_description
FROM log2timeline
WHERE (((date)=#4/27/2012#) AND ((time)>#17:50:52# AND
        (time)<#18:21:8#) AND ((source = "Event Log") AND
        ((file_description) Like "*528*" OR
        (file_description) Like "*538*")) OR
       (((date)=#4/27/2012#) AND ((time)>#17:50:52# AND
        (time)<#18:21:8#) AND ((file_description) Like "*.exe*"))
ORDER BY Date DESC, Time DESC;
```

This query displays the executable programs (`"*.exe*"`) and user login and logoff times (`"*528*"` or `"*538*"`) from 27 April 2012 between the times 17:50:52 and 18:21:08. To expand the artifact listing to display all the artifacts during a given timeframe, the digital forensic professional can use the same query, but remove the section of the statement that displays the executable programs (`(File_Description) Like "*.exe*"`).

5. Test Results

The test data involved a scripted Windows XP incident created with specific activities. In the incident, Miss Scarlet logged into a Windows XP system at 17:50. She opened the Notepad program and then the Calculator program. After closing both programs, she inserted a USB memory stick and ran the program `ProcessList.exe`, which spawned two programs: (i) `ProcessHacker.exe`, which she could see running on the desktop; and (ii) `Services2000.exe`, which she could not see running in the background. `Services2000.exe` corresponds to a Netcat program in listener mode that waits for a hacker to connect to the system. When the Netcat connection is established, a command shell is passed to the hacker allowing him access to Miss Scarlet's Windows XP system. Miss Scarlet left `ProcessHacker.exe` running on the desktop and ran the Solitaire program. She played one game of Solitaire then logged off the system at 18:21.

This incident centers around a file named *HateLetter.txt* containing text that appears to be generated by the user logged in as Miss Scarlet and addressed to Mr. Boddy. The text says that "[she has] had enough of his antics and is going to kill him." The question is whether or not the file was created by Miss Scarlet.

The `log2timeline` tool was executed on a `dd` image of the simulated incident to perform the data collection of temporal artifacts required

Time	Source Type	Description
17:50:55	Event Log	2012 - C:/Program Files/VMware/VMware Tools/vmtoolsd.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:50:55	Event Log	2004 - C:/Program Files/VMware/VMware Tools/VMwareTray.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:50:55	Event Log	580 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:50:55	**Event Log**	**1928 - C:/WINDOWS/explorer.exe - 332 - MissScarlet - (0x0-0x24DEF)**
17:50:55	Event Log	896 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:52:25	Event Log	1436 - C:/WINDOWS/system32/notepad.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:53:08	Event Log	192 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:53:08	Event Log	1568 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:53:11	Event Log	1436 - C:/WINDOWS/system32/notepad.exe - MissScarlet - (0x0-0x24DEF)
17:54:01	Event Log	808 - C:/WINDOWS/system32/calc.exe - 1928 - MissScarlet - (0x0-0x24DEF)
17:54:22	Event Log	808 - C:/WINDOWS/system32/calc.exe - MissScarlet - (0x0-0x24DEF)
18:00:02	Event Log	772 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - (0x0-0x24DEF)
18:00:06	**Event Log**	**908 - /Device/Harddisk1/DP(1)0-0+5/ProcessList.exe - 1928 - MissScarlet - (0x0-0x24DEF)**
18:00:08	**Event Log**	**1612 - C:/DOCUME~1/MISSSC~1/LOCALS~1/Temp/eW_3.tmp/ProcessHacker.exe - 908 - MissScarlet - (0x0-0**
18:00:08	**Event Log**	**1824 - C:/DOCUME~1/MISSSC~1/LOCALS~1/Temp/eW_3.tmp/services2000.exe - 908 - MissScarlet - (0x0-0x2**
18:00:47	Event Log	1992 - C:/WINDOWS/system32/sol.exe - 1928 - MissScarlet - (0x0-0x24DEF)
18:05:54	Event Log	220 - C:/WINDOWS/system32/cmd.exe - 1824 - MissScarlet - (0x0-0x24DEF)
18:14:58	Event Log	1656 - C:/WINDOWS/system32/attrib.exe - 220 - MissScarlet - (0x0-0x24DEF)
18:16:01	Event Log	220 - C:/WINDOWS/system32/cmd.exe - MissScarlet - (0x0-0x24DEF)
18:16:01	Event Log	1824 - C:/DOCUME~1/MISSSC~1/LOCALS~1/Temp/eW 3.tmp/services2000.exe - MissScarlet - (0x0-0x24DEF)
18:20:18	Event Log	1992 - C:/WINDOWS/system32/sol.exe - MissScarlet - (0x0-0x24DEF)
18:20:32	Event Log	908 - /Device/Harddisk1/DP(1)0-0+5/ProcessList.exe - MissScarlet - (0x0-0x24DEF)
18:20:32	Event Log	1612 - C:/DOCUME~1/MISSSC~1/LOCALS~1/Temp/eW 3.tmp/ProcessHacker.exe - MissScarlet - (0x0-0x24DEF)
18:21:06	Event Log	1928 - C:/WINDOWS/explorer.exe - MissScarlet - (0x0-0x24DEF)
18:21:06	Event Log	2004 - C:/Program Files/VMware/VMware Tools/VMwareTray.exe - MissScarlet - (0x0-0x24DEF)

Figure 1. Subset of programs executed during Miss Scarlet's login time.

to construct a super timeline. The `dd` image was mounted with Mount Image Pro version 4. The `log2timeline` tool parsed each artifact to an output file. The `log2timeline` output was imported into a database with the fields shown in Table 1. Several queries were then executed.

The initial queries had a general scope, such as listing the login and logoff times of all users, and when executable programs began and ended execution. The results of these queries provide a digital forensic practitioner with a repeatable starting point for the incident. Because a specific timeframe based on the investigation was set, the super timeline could be narrowed down and the analysis could focus on artifacts that can be correlated in order to determine what happened. When multiple different artifacts, such as the user assist registry key, last accessed time of the file and audit log process begin time, all correlate to the same file and time, then the corresponding process can be confirmed to have executed during a specific timeframe.

5.1 Executing Processes and User Login Timeframe Activity

Based on the incident description, the executing processes and user login timeframe query focused on all programs that executed during Miss Scarlet's login time (i.e., like `.exe`).

The highlighted row in Figure 1 corresponds to when Miss Scarlet's Windows environment began by creating the `Explorer.exe` process

(PID 1928) with parent PID 332. The figure shows that the program `ProcessList.exe` was executed by Miss Scarlet, and that two additional programs were also executed within two seconds (6:00:06 to 6:00:08). This is suspicious behavior because the three programs were executed during a very short period of time, and only `ProcessList.exe` ran from a user directory while the other two programs ran from a temporary directory.

Examination of the three processes in question (highlighted area) reveals that `ProcessList.exe` (PID 908) ran from Miss Scarlet's environment since its parent process is 1928. The two questionable programs, `ProcessHacker.exe` (PID 1612) and `Services2000.exe` (PID 1824) are child processes of `ProcessList.exe` because their parent PID is 908, the PID assigned to `ProcessList.exe`.

More suspicion is raised when the `cmd.exe file` is examined. As highlighted by the arrows, the command prompt began at 18:05:54 creating the event ID 592, and the process ended when event ID 593 was logged at 18:16:01. The command prompt has a parent PID of 1824, which means it was spawned by `Services2000.exe`. All the files and events associated with this ten minute duration that the command prompt ran are now suspect.

The user program activity query reduced the number of entries from 303,000 to 3,000. This saves the digital forensic practitioner from having to sift through a massive number of artifacts. Note that, because Miss Scarlet did not use a browser, web history information would not be present during this time.

5.2 Recent Document Activity

Figure 2 shows the user assist keys from the time `ProcessList.exe` began (18:00:06) through the time the command prompt process ended (18:16:01). The user assist keys are in the registry *HKCU"Software"Microsoft"Windows"CurrentVersion"Explorer"UserAssist*, which logs when a user accesses objects. The arrow points to where `ProcessList.exe` began executing. The bold lines show where the command prompt process began executing. The highlighted line identifies the time when the file `HateLetter.txt` was accessed. This file was created at 18:09 and then modified again at 18:11. Since the file in question, `HateLetter.txt`, was created and modified during the window when the command prompt process was active, and the file has no owner attributes, this file was likely created by the command prompt process and not by Miss Scarlet. Additionally, Miss Scarlet did not run a command prompt nor did she run

Time	SRC	Source Type	Type	Description
18:00:06	**REG**	**UserAssist key**	**Time of Launch**	**UEME_RUNPATH:E:/ProcessList.exe**
18:00:06	EVT	Event Log	Time generated/writt	Security/592;Success;908 - /Device/Harddisk1/DP(1)0-0+5/ProcessList.exe - 1928 - MissSca
18:00:07	FILE	NTFS $MFT	$SI [M..B] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp/ProcessHacker.exe
18:00:07	FILE	NTFS $MFT	$FN [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp/ProcessHacker.exe
18:00:07	FILE	NTFS $MFT	$SI [...B] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp
18:00:07	FILE	NTFS $MFT	$FN [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp
18:00:07	FILE	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/crtdll.dll
18:00:07	FILE	NTFS $MFT	$SI [M.C.] time	/Documents and Settings/MissScarlet/Local Settings/Temp
18:00:08	FILE	NTFS $MFT	$SI [.A..] time	/Documents and Settings/MissScarlet/Local Settings/Temp
18:00:08	FILE	NTFS $MFT	$SI [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp/services2000.exe
18:00:08	FILE	NTFS $MFT	$SI [MAC.] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp
18:00:08	FILE	NTFS $MFT	$SI [.AC.] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp/ProcessHacker.exe
18:00:08	FILE	NTFS $MFT	$FN [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW 3.tmp/services2000.exe
18:00:08	FILE	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/drivers/etc/services
18:00:08	EVT	Event Log	Time generated/writt	Security/592;Success;1612 - C:/DOCUME~1/MISSSC~1/LOCALS~1/Temp/eW 3.tmp/Proce
18:00:08	EVT	Event Log	Time generated/writt	Security/592;Success;1824 - C:/DOCUME~1/MISSSC~1/LOCALS~1/Temp/eW 3.tmp/servic
18:00:42	REG	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/Explorer/MenuOrder/StartMenu2/Programs
18:00:47	FILE	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/sol.exe
18:00:47	REG	NTUSER key	Last Written	Software/Microsoft/Solitaire
18:00:47	EVT	Event Log	Time generated/writt	Security/592;Success;1992 - C:/WINDOWS/system32/sol.exe - 1928 - MissScarlet - VICTIM
18:00:47	REG	NTUSER key	Last Written	Software/Microsoft
18:00:47	REG	UserAssist key	Time of Launch	UEME_RUNPIDL:%csidl2%/Games/Solitaire.lnk
18:00:47	REG	NTUSER key	Last Written	Software/Microsoft/Windows/ShellNoRoam/MUICache
18:00:47	REG	UserAssist key	Time of Launch	UEME_RUNPATH
18:00:47	FILE	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/cards.dll
18:00:47	REG	UserAssist key	Time of Launch	UEME_RUNPIDL:%csidl2%/Games
18:00:47	REG	UserAssist key	Time of Launch	UEME_RUNPATH:C:/WINDOWS/system32/sol.exe
18:00:47	REG	UserAssist key	Time of Launch	UEME_RUNPIDL
18:05:54	**FILE**	**NTFS $MFT**	**$SI [.A..] time**	**/WINDOWS/system32/cmd.exe**
18:05:54	**PRE**	**XP Prefetch**	**Last run**	**CMD.EXE-087B4001.pf: CMD.EXE was executed**
18:05:54	**EVT**	**Event Log**	**Time generated/wri**	**Security/592;Success;220 - C:/WINDOWS/system32/cmd.exe - 1824 - MissScarlet - VICT**
18:06:04	**FILE**	**NTFS $MFT**	**$SI [MAC.] time**	**/WINDOWS/Prefetch/CMD.EXE-087B4001.pf**
18:09:00	**File**	**Dir**		**\Documents and Settings\MissScarlet\My Documents\HateLetter.txt**
18:11:00	File	Dir		\Documents and Settings\MissScarlet\My Documents\HateLetter.txt
18:12:02	FILE	NTFS $MFT	$SI [MAC.] time	/Documents and Settings/MissScarlet/My Documents
18:13:30	FILE	NTFS $MFT	$SI [.A..] time	/RECYCLER/S-1-5-21-1390067357-1078145449-839522115-1005
18:13:30	FILE	NTFS $MFT	$SI [.A..] time	/RECYCLER
18:14:58	FILE	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/ulib.dll
18:14:58	FILE	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/attrib.exe
18:14:58	EVT	Event Log	Time generated/writt	Security/592;Success;1656 - C:/WINDOWS/system32/attrib.exe - 220 - MissScarlet - VICTIM
18:14:58	EVT	Event Log	Time generated/writt	Security/593;Success;1656 - C:/WINDOWS/system32/attrib.exe - MissScarlet - VICTIM - (0x
18:15:41	FILE	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/logon.scr
18:15:41	EVT	Event Log	Time generated/writt	Security/600;Success;628 - C:/WINDOWS/system32/winlogon.exe - VICTIM$ - WORKGROU
18:15:41	EVT	Event Log	Time generated/writt	Security/592;Success;1780 - C:/WINDOWS/system32/logon.scr - 628 - VICTIM$ - WORKGR
18:16:01	EVT	Event Log	Time generated/writt	Security/593;Success;220 - C:/WINDOWS/system32/cmd.exe - MissScarlet - VICTIM - (0x0-
18:16:01	EVT	Event Log	Time generated/writt	Security/593;Success;1824 - C:/DOCUME~1/MISSSC~1/LOCALS~1/Temp/eW 3.tmp/servic

Figure 2. Query results for the `ProcessList.exe` and `cmd.exe` processes.

Notepad to create file `HateLetter.txt` during this timeframe, placing even more suspicion on the command prompt process.

5.3 Summary

A variety of simple and complex queries can be created based on the evidence collected for an incident. The queries discussed above demonstrate how evidence can be quickly discovered using simple queries and how a practitioner can progress through a more complex, yet granular, examination of the artifacts. These queries enable the practitioner to reduce hundreds of thousands of artifacts to roughly one hundred artifacts or less, helping focus the investigation on the artifacts of interest.

6. Conclusions

Queries of the output of a tool such as `log2timeline` can be used very effectively to produce super timelines of events in incidents involving Windows systems. The super timelines provide excellent overviews

of the events that occurred before, during and after the incidents of interest. Four reusable queries were presented for extracting concise timeline artifacts that allow a digital forensic professional to decisively determine the activities involved in incidents. The queries can dramatically decrease the number of artifacts that need to be examined by digital forensic professionals – in the simulated incident, the number of artifacts was reduced from 303,000 to 100.

The approach presented in this paper does not incorporate auto correlation methods that could be used to fuse multiple timestamps associated with a single event; this is an important area for future exploration because of the variety of timestamps with different temporal resolutions that must be accounted for when creating a super timeline. Another area for future research involves the creation of an interactive visualization of the `log2timeline` output that would provide a means to view details at multiple levels of abstraction.

The views expressed herein are those of the authors and do not reflect the official policy or position of the U.S. Air Force, U.S. Department of Defense or the U.S. Government.

References

[1] B. Carrier, The Sleuth Kit (`www.sleuthkit.org`).

[2] H. Carvey, RegRipper (`regripper.wordpress.com`).

[3] J. Foster and V. Liu, Timestomp (`www.forensicswiki.org/wiki/Timestomp`).

[4] K. Guojonsson, Mastering the Super Timeline with `log2timeline`, SANS Gold Paper, SANS Institute, Bethesda, Maryland, 2010.

[5] K. Jones, Pasco v.1.0, McAfee, Santa Clara, California (`www.mcafee.com/us/downloads/free-tools/pasco.aspx`), 2012.

[6] Mandiant, Web Historian, Alexandria, Virginia (`www.mandiant.com/resources/download/web-historian`).

[7] J. Olsson and M. Boldt, Computer forensic timeline visualization tool, *Digital Investigation*, vol. 6(S), pp. S78–S87, 2009.

[8] SIMILE Project, The JFK assassination timeline with Dutch timeline labels, Massachusetts Institute of Technology, Cambridge, Massachusetts (`www.simile-widgets.org/timeline/examples/jfk_i18n/jfk.html`), 2009.

[9] TZWorks, Yet Another Registry Utility (`yaru`), Herndon, Virginia (`www.tzworks.net/download_links.php`).

Chapter 10

LOG FILE ANALYSIS WITH CONTEXT-FREE GRAMMARS

Gregory Bosman and Stefan Gruner

Abstract Classical intrusion analysis of network log files uses statistical machine learning or regular expressions. Where statistically machine learning methods are not analytically exact, methods based on regular expressions do not reach up very far in Chomsky's hierarchy of languages. This paper focuses on parsing traces of network traffic using context-free grammars. "Green grammars" are used to describe acceptable log files while "red grammars" are used to represent known intrusion patterns. This technique can complement or augment existing approaches by providing additional precision. Analytically, the technique is also more powerful than existing techniques that use regular expressions.

Keywords: Intrusion detection, log file analysis, context-free grammars

1. Introduction

Most modern intrusion analysis systems rely on pattern recognition to differentiate malicious network traffic from benign traffic [1, 4, 6]. Anomaly-based detection systems are designed to recognize traffic generated during normal network operations. These systems use information such as source and destination addresses and ports, applications that generate traffic, and other header information to determine whether or not a packet should raise an alarm. Sometimes, the analysis proceeds one step further, scrutinizing the packets as a stream instead of just packet by packet. Signature-based analysis systems operate similarly and, in some cases, search for signatures of injected code in network packets. The search typically attempts to identify a series of bits that match certain regular expressions.

State-based analysis is a promising security approach [7, 10]. Gudes and Olivier [7] emphasize that the state of an application should also be

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 145–152, 2013.

considered in assessing the security of an application. They reason that certain messages received by an application, while perfectly safe in one state, may compromise the security of the application should the messages be received in some other, more vulnerable state. Therefore, the state of an application should be taken into account when determining whether or not incoming data is harmful. Gudes and Olivier [7] proposed the use of context-free grammars for state-based analysis. Their approach entails the construction of a grammar to represent the transitions from state to state internally within an application; the grammar also describes the actions that are allowable in any given state. This paper presents an enhancement of the context-free grammar approach for the purpose of intrusion detection.

2. Related Work

Memon [5] has employed context-free grammars in a log file categorization system. Memon developed a grammar capable of representing a single packet in a data stream, including information such as packet protocol, source and destination. Also, a method of grammar inference was developed, which is capable of expanding an existing grammar modeled to identify benign packets at the discretion of an administrator. However, Memon's approach does not consider the sequencing of packets in a data stream, nor does it take into account the "statefulness" of an interaction between network applications. Also, a grammar that could capture the statefulness of applications communicating over a network would be considerably more complex than the grammars used by Memon.

The possibility of dealing with too many false alerts in automated intrusion detection systems was addressed by Harang and Guarino [3]. They used various heuristics to condense a massive amount of elementary alerts into a manageable number of meta-alerts. However the alert condensation technique has limited theoretical underpinnings. Similar work by Valdez and Skinner [9] resulted in a fusion system that groups alerts based on their similarity (near-matches) to existing meta alerts.

3. Method

The context-free grammar described in this paper handles a subset of malicious attack vectors. The grammar, which fits well into the digital forensics context, is related to formal methods in the field of model-based testing [2] and may be regarded as a limited form of automated verification on the basis of specific samples. A proof goal is a decision whether or not a network intrusion attempt has occurred in a specific situation.

```
TRACE      -> e | TRANSF | ADRESO | DNSQRY
TRACED1    -> TRANSFD1 | ADRESOD1 | DNSQRY1
TRACET1    -> TRANSF1  | ADRESOT1 | DNSQRYT1
TRACET2    -> TRANSF2  | ADRESOT2 | DNSQRYT2
TRACET3    -> PDU      | ADRESOT3 | DNSQRYT3
TRACET1D1  -> TRANSF1D1 | ADRESOT1D1 | DNSQRY1T1
TRACET2D1  -> TRANSF2D1 | ADRESOT2D1 | DNSQRY1T2
TRACET3D1  -> PDUD1  | ADRESOT3D1 | DNSQRY1T3
TRACEP1    -> PDU1   | ADRESOP1   | DNSQRYP1
TRACEP2    -> PDU2   | ADRESOP2   | DNSQRYP2
TRACEP3    -> PDU3   | ADRESOP3   | DNSQRYP3
TRACED1P1  -> PDU1D1 | ADRESOD1P1 | DNSQRY1P1
TRACED1P2  -> PDU2D1 | ADRESOD1P2 | DNSQRY1P2
TRACED1P3  -> PDU3D1 | ADRESOD1P3 | DNSQRY1P3
TRANSF     -> syn TRACET1
TRANSF1    -> synack acko get TRACET2
TRANSF2    -> acki TRACET3
TRANSFD1   -> syn TRACET1D1
TRANSF1D1  -> synack acko get TRACET2D1
TRANSF2D1  -> acki TRACET3D1
ADRESO     -> arpq arpr TRACE
ADRESOT1   -> arpq arpr TRACET1
ADRESOT2   -> arpq arpr TRACET2
ADRESOT3   -> arpq arpr TRACET3
```

Figure 1. Green grammar.

Two types of grammars are used to tackle the problem effectively from both sides:

- **Green Grammar:** One green grammar is used to describe log files corresponding to harmless communications. A log file that is successfully parsed with a green grammar is regarded as "hypothetically harmless."
- **Red Grammars:** Several red grammars are used to describe log files that contain traces corresponding to known malicious communications patterns. A log file that is successfully parsed with a red grammar is regarded as "harmful" and raises an alert. A log file that cannot be parsed with a red grammar is regarded as "hypothetically harmless."

Figures 1 and 2 present the green grammar that represents acceptable traffic. The grammar was generated from experimental traffic and log files created during the use of a web browser. Each terminal symbol in the grammar represents a single packet that was recorded. The grammar as a whole models a string of packets of a "conversation" that occurs

```
ADRESOT1D1 -> arpq arpr TRACET1D1
ADRESOT2D1 -> arpq arpr TRACET2D1
ADRESOT3D1 -> arpq arpr TRACET3D1
ADRESOP1   -> arpq arpr TRACEP1
ADRESOP2   -> arpq arpr TRACEP2
ADRESOP3   -> arpq arpr TRACEP3
ADRESOD1P1 -> arpq arpr TRACED1P1
ADRESOD1P2 -> arpq arpr TRACED1P2
ADRESOD1P3 -> arpq arpr TRACED1P3
DNSQRY     -> dnsq TRACED1
DNSQRY1    -> dnsr TRACE
DNSQRYT1   -> dnsq TRACET1D1
DNSQRYP1   -> dnsq TRACED1P1
DNSQRY1T1  -> dnsr TRACET1
DNSQRY1P1  -> dnsr TRACEP1
DNSQRYT2   -> dnsq TRACET2D1
DNSQRYP2   -> dnsq TRACED1P2
DNSQRY1T2  -> dnsr TRACET2
DNSQRY1P2  -> dnsr TRACEP2
DNSQRYT3   -> dnsq TRACET3D1
DNSQRYP3   -> dnsq TRACED1P3
DNSQRY1T3  -> dnsr TRACET3
DNSQRY1P3  -> dnsr TRACEP3
PDU        -> data TRACEP1 | http TRACEP3
PDU1       -> data TRACEP2 | http TRACEP3
PDU2       -> acko TRACET3
PDU3       -> acko TRACE
PDUD1      -> data TRACED1P1 | http TRACED1P3
PDU1D1     -> data TRACED1P2 | http TRACED1P3
PDU2D1     -> acko TRACET3D1
PDU3D1     -> acko TRACED1
```

Figure 2. Green grammar (continued).

when a web page is transferred. Ideally, the conversation follows the simplified pattern:

```
TRACE -> syn synack acko get acki PDU
PDU   -> data data acko PDU
       | data http acko
       | http acko
```

A communication involving a browser begins with a handshake, a sequence of packets with synchronization flags or acknowledgement flags or both flags set. Next, a request is made for data transfer, a GET followed by a packet in which the acknowledgement flag is set (again). Data transfer then begins. This is represented by the protocol data unit (PDU) section. The rest is the data transfer of two packets at a time until

Table 1. Green grammar results.

Input	Total	Accepted	Rejected	Lex Err	Syn Err
Benign	100	94	6	6	0
Malicious	100	0	100	28	72

the transfer is complete. After every two packets, an acknowledgement is sent to indicate there are no errors and that it is possible to continue with the transfer. When there is no more data, a final HTTP packet is received to indicate the end of the stream, followed by one last outgoing acknowledgement to the web server.

The example pattern is an ideal model of a conversation between a client and server. In practice, however, streams tend to be interspersed with additional mini-conversations that carry meta information. For example, an initial Address Resolution Protocol (ARP) message may be sent in order to find the location of a web page. Also, in nearly every trace, there is at least one request for a Domain Name Server (DNS), which tends to jump straight into the middle of the main conversation with the web server. Therefore, an appropriate green grammar must be constructed such that it retains the internal state of a conversation when it is interrupted and returns to the original state after the interrupt.

This implies that an entire conversation must complete in order to be recognized as a benign trace. If the conversation starts, it must terminate successfully. If it does not, then the browser may have been compromised (unless the communication was technically interrupted, for example by a browser crash on the client side). In such a case, the parsing of the corresponding trace leads to a syntax error and the trace is classified as "suspicious."

It is worth noting that the mini-ARP-conversation is always completed without an interruption. However, as in the case of a web page request, a DNS request is susceptible to interruption. Unfortunately, regular expressions (unlike the more expressive context-free grammars) are inadequate to remember the status of a conversation before the occurrence of an interrupt.

4. Experimental Results

We conducted an experiment that used the green grammar in a controlled laboratory network setting with various permissible and non-permissible input strings. ANTLR [8] was used to generate the parser for the green grammar. Table 1 summarizes the experimental results.

Two types of errors are of interest in the analysis:

- **Alpha Error (False Positive):** This error is the incorrect rejection of benign input (6/100 = 6% from Table 1). An alpha error is annoying or inconvenient, but it is not harmful.
- **Beta Error (False Negative):** This error is the incorrect acceptance of malicious input (0/100 = 0% from Table 1). A beta error is interpreted as being harmful.

Despite the zero beta error, a problem with the grammar is that not all parsing failures resulted from syntax errors, which would be the ideal case if the grammar was *a priori* aware of all possible input tokens. The non-zero lexical errors (errors prior to the parsing phase) show that an input stream can contain unknown symbols that the grammar does not recognize. Thus, the suitability (fit-for-purpose quality) of the intrusion detection grammar can be estimated as follows:

- In the ideal alpha case, no parsing errors should occur. Since all the errors occurred during the lexical analysis phase, 100% percent of the alpha errors are from not recognizing all the symbols in the input strings.
- In the ideal beta case, only parsing errors should occur. Only 72% percent of the (correct) rejection decisions were based on parsing whereas 28% percent of all rejections were based on "decisionless" lexical errors before the decision-making parsing phase. Therefore, the beta errors are estimated to be as high as 28%.

We conjecture that the errors appear in the context of the occasional mini-communications, which can be corrected for by modifying the manner in which the grammar is processed.

The experiments with malicious input streams were restricted to scenarios in which communications terminate mid-stream, as in the case of buffer overflow attacks. The parser should correctly recognize the absence of stream-closing data packets and label the stream as suspicious. However, in one case, the lexical analyzer could not recognize a termination symbol that was sent properly to terminate a conversation. This problem might be linked to speed discrepancies between data download and analysis. Specifically, the parsing unit may prematurely believe that an input stream has been "hacked" while in fact a harmless communication is in progress.

5. Discussion

Because context-free languages are a superclass of regular expressions and are, therefore, more complex, they have greater computational re-

quirements. A critical question is whether the additional complexity provided by context-free grammars renders the approach impractical. To answer this question, more examples that cannot be treated with regular expressions must be identified. The speed of analysis is the main reason for using regular expressions with Cisco networking devices because the analysis of data streams proceeds in an online and on-the-fly manner. Regular expression analysis can be implemented in a runtime-efficient manner. However, *ex post facto* log file analysis does not have strict real-time processing requirements.

A practical problem is posed by interrupted communication attempts, for example, due to a crashed or frozen browser on the client side of an Internet connection. On the server side, this results in a broken trace in the log file that cannot be parsed, regardless of whether the interrupted communication is malicious or harmless. Traces of interrupted communication attempts could increase the rate of alpha errors. To tackle this problem, a method is needed to divide a large trace into shorter sub-traces that could be parsed individually. Alternatively, the grammar could be modified so that interrupted communications are explicitly identified as such.

6. Conclusions

The application of context-free grammars to describe acceptable log files and intrusion patterns can significantly enhance log file analysis. The resulting approach is more powerful and provides better precision than existing techniques based on regular expressions.

Our future research will focus on developing a software framework that enables users to generate case-specific grammars and conduct detailed analyses. We will also attempt to cast a large number of well-known intrusion patterns as case-specific red grammars. A suitable combination of green and red grammars would reduce the alpha and beta error rates; however, this would require an additional meta decision policy for cases where the decisions made by the green and red parsers are inconsistent.

Acknowledgement

We thank Bruce Watson for his assistance with the regular expression technique in the context of Cisco networking equipment. We also thank the anonymous reviewers and the conference attendees for their insightful critiques and comments.

References

[1] S. Axelsson, Intrusion Detection Systems: A Survey and Taxonomy, Technical Report, Department of Computer Science, Chalmers University, Goteborg, Sweden, 2000.

[2] S. Gruner and B. Watson, Model-based passive testing of safety-critical components, in *Model-Based Testing for Embedded Systems*, J. Zander, I. Schieferdecker and P. Mosterman (Eds.), CRC Press, Boca Raton, Florida, pp. 453–483, 2011.

[3] R. Harang and P. Guarino, Clustering of Snort alerts to identify patterns and reduce analyst workload, *Proceedings of the 2012 Military Communications Conference*, 2012.

[4] T. Lunt, A survey of intrusion detection techniques, *Computers and Security*, vol. 12(4), pp. 405–418, 1993.

[5] A. Memon, Log File Categorization and Anomaly Analysis Using Grammar Inference, M.S. Thesis, School of Computing, Queen's University, Kingston, Canada, 2008.

[6] P. Ning and S. Jajodia, Intrusion detection techniques, in *The Internet Encyclopedia, Volume 2*, H. Bidogli (Ed.), Wiley, Hoboken, New Jersey, pp. 355–367, 2004.

[7] M. Olivier and E. Gudes, Wrappers: A mechanism to support state-based authorization in web applications, *Data and Knowledge Engineering*, vol. 43(3), pp. 281–292, 2002.

[8] T. Parr, *The Definitive ANTLR Reference: Building Domain-Specific Languages*, Pragmatic Bookshelf, Raleigh, North Carolina, 2007.

[9] A. Valdez and K. Skinner, Probabilistic alert correlation, *Proceedings of the Fourth International Symposium on Recent Advances in Intrusion Detection*, pp. 54–68, 2001.

[10] S. Zhang, T. Dean and S. Knight, A lightweight approach to state-based security testing, *Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research*, article no. 28, 2006.

Chapter 11

USING A GOAL-DRIVEN APPROACH IN THE INVESTIGATION OF A QUESTIONED CONTRACT

Clive Blackwell, Shareeful Islam and Benjamin Aziz

Abstract This paper presents a systematic process for describing digital forensic investigations. It focuses on forensic goals and anti-forensic obstacles and their operationalization in terms of human and software actions. The paper also demonstrates how the process can be used to capture the various forensic and anti-forensic aspects of a real-world case involving document forgery.

Keywords: Forensic investigation, goal-driven process, questioned documents

1. Introduction

There is an acute need to extend the typical digital forensic investigation process to handle complex cases involving large quantities of data from multiple computers and devices. The investigation process must cope with the many difficulties inherent in evidence collection and analysis from intentional and deliberate causes that may result in evidence being incorrect, incomplete, inconsistent or unreliable.

Many of the existing digital forensic investigation processes emphasize collecting evidence or directly starting with the incident. The processes generally involve steps such as collecting, preserving, examining, analyzing and presenting digital evidence [14–16, 19]. In addition, many investigations are bottom-up, focusing on the collection and analysis of data using an exhaustive search of the media based on keywords and regular expressions. However, it is often infeasible to examine all the supplied media. Also, as Casey and Rose [5] emphasize, it may be ineffective – vital evidence is often missed because there are no matches for low-level patterns.

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 153–167, 2013.

A systematic reasoning process for analyzing digital forensic investigation requirements is currently unavailable. Additionally, a forensic investigation should address problems posed by anti-forensics, especially when time, cost and resources are critical constraints in the investigation. To address these needs, this paper presents a goal-driven methodology to specify the requirements of a digital forensic investigation. The proposed systematic process initiates with the identification of the main goals of the investigation and the analysis of the obstacles that could hinder the goals. The process integrates the anti-forensics dimension within the digital forensic investigation process at the level of requirements that overcome the deliberate obstacles. In this way, the methodology supports existing forensic processes by offering a systematic investigation strategy to manage evidence so that it helps attain the investigative goals and overcome the technical and legal impediments in a planned manner.

Many formal methodologies have been proposed for requirements engineering and analysis, including i*/Tropos [7] and KAOS [20]. Our approach follows KAOS in line with existing work [1]. According to Leigland and Krings [13], adopting a formal and systematic approach has several benefits: (i) procedural benefits by reducing the amount of data and aiding their management; (ii) technical benefits by allowing digital forensic investigations to adapt to technological changes; (iii) social benefits by capturing the capabilities of the perpetrators within the social and technical dimensions; and (iv) legal benefits by allowing the expression of the legal requirements of forensic investigations.

The systematic process is demonstrated on a recent case involving alleged document forgery and questionable claims made by Paul Ceglia against Mark Zuckerberg of Facebook [18]. The analysis helps outline the main obstacles to the various claims and evidence in the case. Also, it proposes how the requirements underlying the claims and evidence are operationalized by means of investigator activities along with forensic system and software operations.

2. Related Work

Several researchers have discussed the forensic investigation process and techniques relating to anti-forensics. This section presents a brief overview of the approaches relevant to this work.

Kahvedzic and Kechadi [11] have presented a digital investigation ontology as an abstraction of concepts and their relationships for the representation, reuse and analysis of digital investigation knowledge. The ontology is based on four dimensions: crime case, evidence location, in-

formation and forensic resource. The approach has been used to model knowledge about the Windows registry.

Reith, *et al.* [16] have proposed an abstract digital forensics model comprising components such as the identification, preparation, analysis, presentation and return of evidence. The model supports future digital technologies and understanding by non-specialists.

Hunton [10] has used utility theory for cyber crime execution and analysis. The work shows that law enforcement officers could leverage cyber crime execution and analysis models when investigating crimes to support the analysis of the evidence regardless of the level of complexity of the crime.

Carrier and Spafford [4] consider a computer or other digital device involved in a crime as a digital crime scene. They employ a process model for forensic investigations that comprises five phases: readiness, deployment, physical/digital crime scene investigation and presentation. Huber, *et al.* [9] have presented techniques for gathering and analyzing digital evidence from online social networking sites.

Harris [8] has discussed techniques for destroying, hiding and eliminating evidence resources as part of anti-forensic activities. Dahbur and Mohammad [6] identify time, cost, forensic software vulnerabilities, victim privacy and the nature of the digital evidence as the main challenges posed by anti-forensic activities.

Several of the works mentioned above focus on systematic forensic investigation processes with an emphasis on collecting and analyzing evidence. However, the systematic process presented in this paper stands out because it combines forensic and anti-forensic considerations in a single investigative framework.

3. Proposed Process

Figure 1 shows the proposed systematic process. The process begins with understanding the investigation processes starting with the incident context analysis and ending with the appropriate actions for analyzing the evidence.

The systematic process considers anti-forensic issues during the forensic investigation process so that possible obstructions can be identified, analyzed and overcome. The process consists of four activities that define the major areas of concern. Each activity incorporates steps concerning the creation of artifacts such as goals, obstacles, evidence and forensic actions relating to the incident. The artifacts are incrementally combined to produce the incident report containing textual and graphical representations. The process defines the roles that take responsibility for

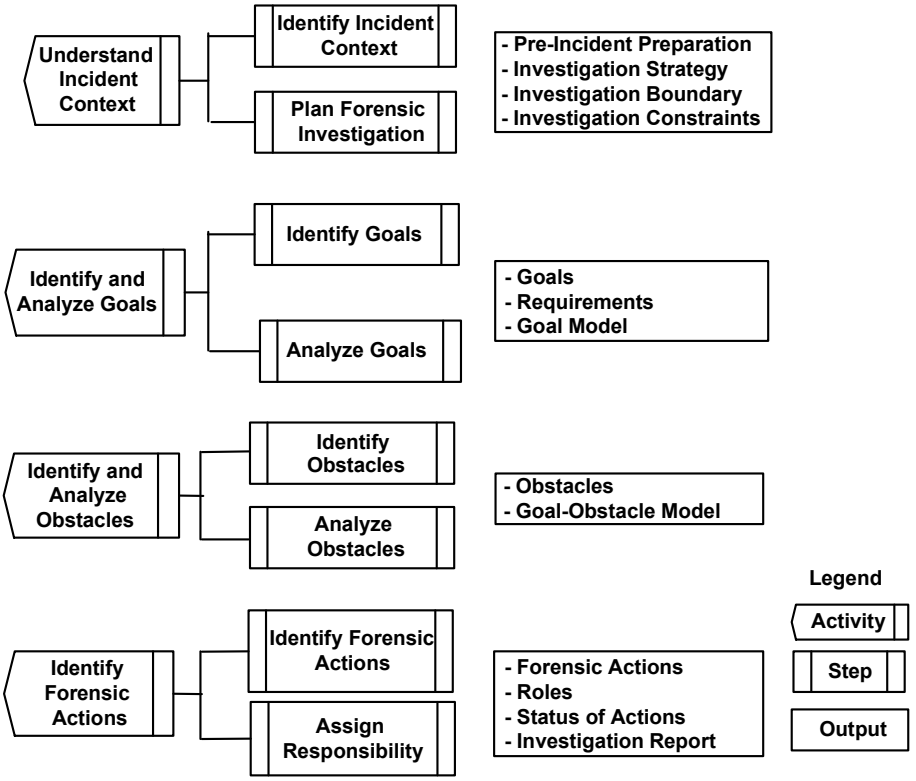


Figure 1. Systematic process for digital forensic investigations.

specific artifacts and perform actions that produce or modify artifacts. The activities are performed sequentially and, if necessary, a number of iterations of individual activities may be performed until they are completed.

3.1 Activity 1: Understand Incident Context

The first activity is to understand the background of the incident. This includes pre-incident preparation, choosing the investigation team, determining the investigation strategy, discovering the complexity and severity of the incident, and establishing the boundary of the forensic process. After the incident context is identified, the forensic team formulates a plan for performing the investigation. This involves choosing a strategy to isolate, secure and preserve the state of the physical and digital evidence. The plan should consider the investigation constraints such as media size, time and budgetary restrictions, and availability of resources such as tools, equipment and expertise.

3.2 Activity 2: Identify and Analyze Goals

After the incident context is defined, the next activity is to identify and model the goals of the forensic investigation. Forensic investigations have primary goals such as conducting a successful investigation and collecting and preserving evidence. The explicit determination of the goals aids in the justification and delineation of the scope of the investigation. The goals may also include suggesting investigative leads and abandoning fruitless leads, as well as proving cases by discovering and overcoming potential weaknesses.

The next step in the activity is to analyze the identified goals so that the higher-level goals are refined into sub-goals. In particular, this step considers how the various phases of the investigation process link the sub-goals with the main goal and support incident analysis. For example, collecting evidence as a goal can be refined to gathering evidence from different systems, devices and the Internet, possibly from unusual locations. In the Ceglia case [18], information was found that pointed to other locations where important evidence might be located, such as undisclosed email accounts and third party systems belonging to Ceglia's lawyers.

The sub-goals may be linked by AND or OR refinement relations to construct the goal model. An AND refinement specifies that all the sub-goals must be satisfied for the parent goal to be satisfied, while an OR refinement specifies that any one of the sub-goals is sufficient to satisfy the parent goal [20].

3.3 Activity 3: Identify and Analyze Obstacles

Obstacles hinder the ability to achieve the goals. Therefore, obstacle identification and analysis focus on what could go wrong during a forensic investigation, specifically with regard to evidence collection, preservation and analysis. It is necessary to identify all the plausible obstacles to determining the facts about an incident. Determining the obstacles in advance facilitates the selection of a course of action to overcome them.

This step assesses the potential damage to the overall investigation caused by obstacles. These include difficulties in finding evidence, exhausting the anticipated time and resources, dealing with the manipulation of essential metadata such as hashes and timestamps, and storing data anonymously on the Internet rather than locally.

Generally, the evidence should be admissible (must be able to be used in court), authentic (original and unchanged), reliable (correct and accurate), complete (all relevant evidence is available) and believable (easy to understand and credible to a jury). An obstacle can affect the

integrity, completeness, reproducibility, timeliness and believability of a forensic activity as well as the outputs of the activity. Obstacle analysis focuses on understanding the types of obstacles posed by anti-forensic actions.

3.4 Activity 4: Identify Forensic Actions

The final activity of the process is to identify the appropriate forensic actions that must be applied based on the critically of the incident. These actions operationalize goal satisfaction to determine a suitable response strategy to resolve the incident. In order to choose the appropriate actions, it is necessary to understand the risks due to the occurrence of the incident and the obstacles posed by anti-forensic activities. Risk has various dimensions such as financial loss, loss of reputation and privacy, and intellectual property theft. Before choosing the actions, it is necessary to consider the legal constraints regarding notifications to regulatory authorities and the quality of the documentation of the investigative goals and requirements.

Consider for example the Ceglia v. Zuckerberg and Facebook case discussed below, which examined the authenticity of a business contract and the supporting evidence such as relevant emails. An obstacle would exist if a copy rather than the original contract were to be provided and the supporting evidence suggests that it could be authentic. Hence, the forensic actions would focus on the use of low-level tools to find anomalies in metadata and timestamps pertaining to the copy.

The selected forensic actions should be implemented to successfully complete the investigation. Also, the effectiveness of the implemented control actions should be monitored.

4. Case Study

The application of the systematic process is demonstrated using a civil case filed in 2010 by Paul Ceglia against Mark Zuckerberg and Facebook.

According to the complaint, Paul Ceglia, an entrepreneur, engaged Mark Zuckerberg to perform some work on his StreetFax Project around the time that Zuckerberg founded Facebook in 2003. Ceglia paid Zuckerberg $1,000 for work on StreetFax and also claimed that he paid $1,000 to fund Zuckerberg's "face book project." He produced a work for hire contract that was apparently signed by himself and Zuckerberg covering the two projects [2]. According to Ceglia, the agreement stated that Ceglia would get 50% of the "face book project" in exchange for funding its initial development. Zuckerberg clearly discussed Facebook with

Ceglia; this fact was supported by multiple email exchanges between the two parties.

The court ordered Mr. Ceglia to produce relevant electronic assets such as an electronic copy of the contract, copies of the purported emails, and the computers and electronic media under Ceglia's control. The court also issued an electronic asset inspection protocol for inspecting the collected evidence, requesting that the investigators check the authenticity and availability of the evidence and provide a report to the court.

The digital forensic analysis was provided in an expert report by Stroz Friedberg [18] for Zuckerberg, which was made public after it was submitted to the court. Several expert reports related to the physical evidence were also provided, especially those by LaPorte [12] and Romano [17].

4.1 Activity 1: Understand Incident Context

This activity focuses on understanding the issues related to the investigation. The main scope of the investigation is to confirm the authenticity of the claims submitted by Ceglia pertaining to the work for hire contract and purported emails, including checking the timestamps and formats of the collected evidence. In addition, the evidence has to be forensically sound to support the electronic asset inspection protocol and it should be possible to identify if any of the evidentiary materials are forgeries.

A crucial first step is to acquire all of Ceglia's computer equipment and other devices that he used in his dealings with Zuckerberg, such as his parents' computer that was found to contain the original contract, and to discover and preserve evidence from his online activities, including his use of multiple email accounts. The complexity of the investigation mainly arises from the quantity of electronic data from different geographical locations and the need to preserve and check all the possible evidence. The digital evidence was contained in three hard drives, 174 floppy disks and 1,087 CDs. The relevant evidence was in the form of image files, email communications, and draft and deleted documents. Appropriate skills and tools exist for the investigation, and we do not consider issues such as investigation team management and time and budget in this case study.

4.2 Activity 2: Identify and Analyze Goals

The goal of the defense in the Ceglia case is to prove that the work for hire contract is a forgery. This would result in the failure of Ceglia's

claim of part ownership of Facebook because it is the only evidence available that could prove Ceglia's version of the events. The main goal, as shown in Figure 2, can be refined into sub-goals related to the production and analysis of all the relevant computer and electronic media, including the purported contract and email correspondence.

A generic goal tree for document forgery developed for similar cases can help determine an initial approach that focuses attention on the likely evidence and its potential locations. The goal tree has three branches to demonstrate the invalidity of the work for hire document, and an additional branch to show that the case fails on technical grounds due to withheld or spoiled evidence.

In theory, it is sufficient to prove forgery in only one way, so the goal is an OR refinement of the four possibilities shown in Figure 2. However, the proof of forgery should be considered in multiple ways to ensure that the case is resilient to unanticipated new evidence and legal challenges.

We decomposed all four branches of the goal tree, but choose to explain only the most convincing branch that makes the fewest assumptions by directly attempting to show that the contract is a forgery. This is adequate because Ceglia's case would fail because the purported contract is the only compelling evidence for his claim.

4.3 Activity 3: Identify and Analyze Obstacles

Several obstacles impede the goal of the investigation to show that the work for hire contract is a forgery. Obstacles to a direct proof of forgery are the lack of original documents; only copies are available to support the contract. Figure 3 shows the goal tree for overcoming the obstacles in this branch. The obstacles slope in the opposite way than do goals, are colored gray and have dotted borders. The figure also shows that further goals overcome many of the obstacles as children of the obstacle nodes, but any obstacle without a child goal node is not surmounted. The evidence is convincing in this case. However, in other cases of alleged document forgery, the obstacles to direct proof may be considerable. This may require other branches that provide weaker substantiation to be investigated instead (Figure 2).

The two primary pieces of evidence supplied by Ceglia are the alleged work for hire contract and supporting emails. There is the apparent authenticity of the contract based on its content, and the supporting emails appear to give a consistent account that supports Ceglia's version of the events. An important obstacle to proving forgery is that the original contract and supporting email messages are not available. Therefore, the investigation has to rely on secondary evidence from deleted and

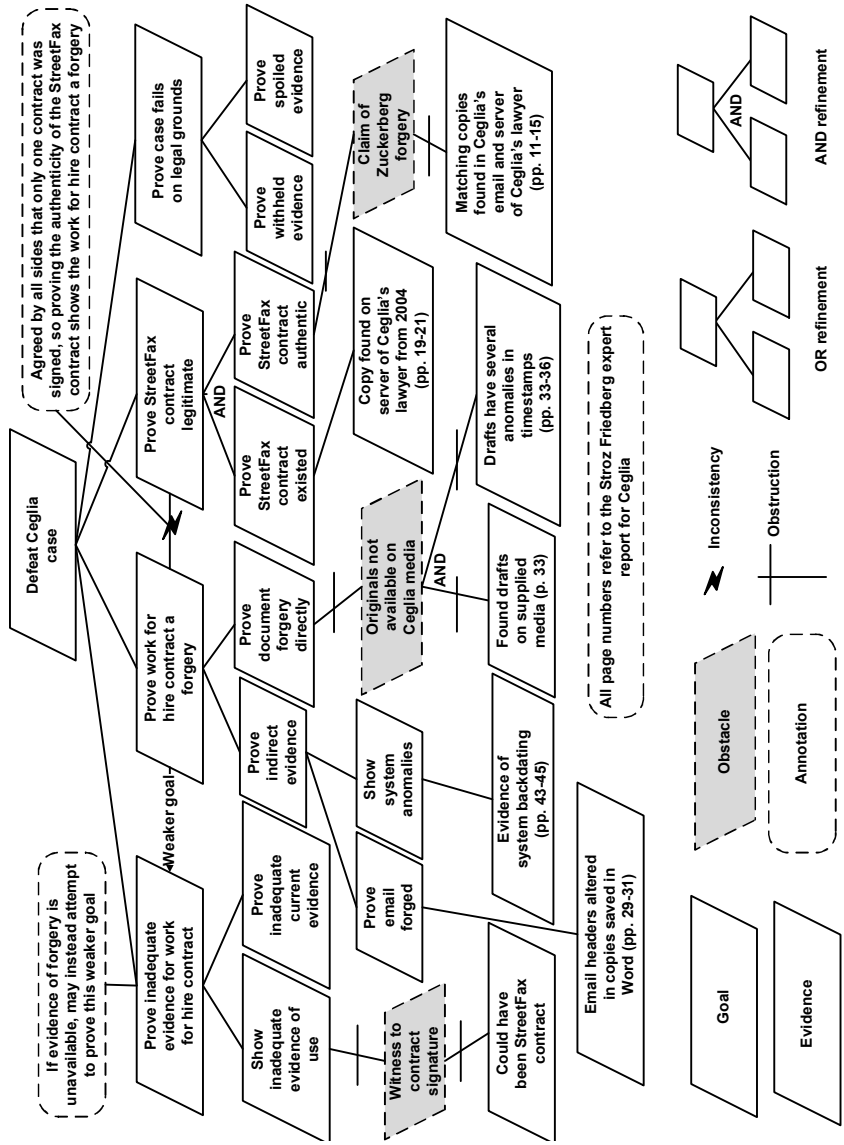


Figure 2. Ceglia case overview.

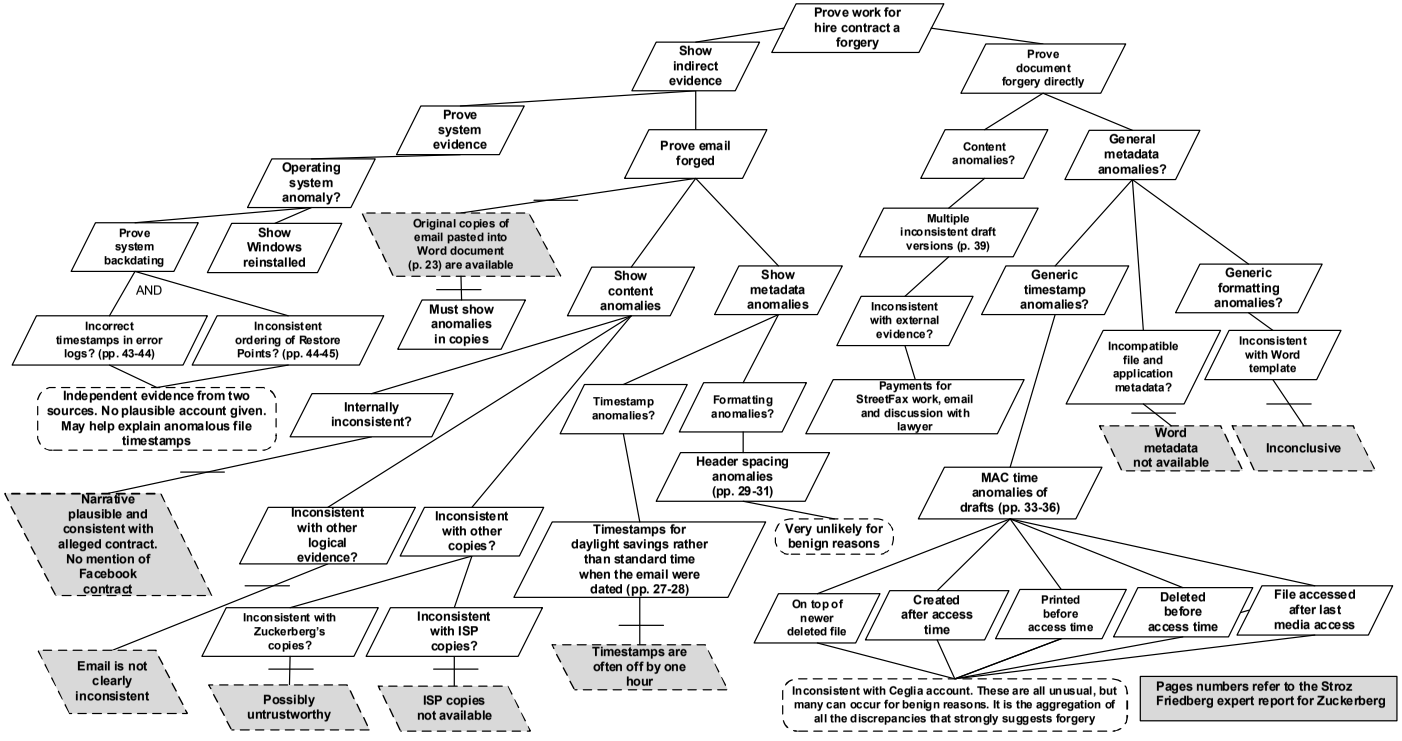


Figure 3. Decomposition of part of the goal tree to prove that the work for hire contract is a forgery.

draft contract files, and email messages that were cut and pasted into a Word document. However, the purported emails have formatting discrepancies in their headers that are inconsistent and indicate that the emails were manually typed and edited after being copied to the Word document. There is evidence of possible spoliation, especially due to multiple re-installations of the Windows operating system and the deletion and overwriting of relevant files. Therefore, the obstacles in this case are mainly unverified and incomplete evidence, with the primary evidence being unavailable because no exact copies of the work for hire document were found on the media.

4.4 Activity 4: Identify and Execute Actions

The forensic actions need to achieve sufficient goals and overcome obstacles to achieve the primary goal of showing that the work for hire contract is a forgery. Most of the nodes are OR branches, so only one path with sufficient evidence from a leaf node to the root is necessary, and there are always alternatives for bypassing insurmountable obstacles. However, as mentioned above, it is safer to prove a case in multiple ways. Therefore, each branch of the primary goal tree in Figure 2 is decomposed to prove the case in four different ways.

In the first branch, there is no independent evidence for the work for hire contract, except for the eyewitness who witnessed a contract signature, but the StreetFax contract has better provenance and it is more likely that it was in fact signed. The third branch contains convincing evidence for the authenticity of the StreetFax contract, which shows the work for hire contract to be a forgery, because there was only one contract between the two parties. In his expert report for Ceglia, Broom [3] gave an alternative hypothesis that Zuckerberg or his agents could have forged the StreetFax contract. However, this is convincingly refuted by the discovery of the StreetFax contract independently in Ceglia's email and on a server belonging to Ceglia's lawyer from 2004, six years before the case was filed [18]. The fourth branch checks for evidence spoliation and evidence withholding. The evidence includes the deletion of relevant files such as the StreetFax contract and draft work for hire documents, the deletion of email messages and the deactivation of email accounts in an apparent attempt to avoid discovery. Multiple operating system re-installations that overwrote the data on the hard disk is evidence of spoliation, but this could also have an innocent explanation.

The second branch demonstrates the evidence that the work for hire contract is a forgery (Figure 3). Although the content appears plausible, the metadata provides evidence of forgery. Several actions lead to

convincing evidence, including checking for inconsistency in email messages. The emails give a plausible account and support. Additionally, the physical tests demonstrate beyond reasonable doubt that the work for hire contract was created using a fake page 2 attached to the legitimate page 1 from the StreetFax contract. This was demonstrated in multiple ways by experts, especially LaPorte [12] and Romano [17], who showed that different toners and inks were used on the two pages. We do not discuss this further because it is outside the domain of digital forensics.

The artifacts produced from the previous activities are incrementally combined to produce the forensic investigation report. The report should also include the status of the implemented forensic actions and their effectiveness. Although, the expert report for Zuckerberg by Stroz Friedberg [18] was comprehensive and highlighted all the relevant points, a more systematic exposition of the overall argument would have provided a clearer narrative.

5. Discussion

The Caglia v. Zuckerberg and Facebook case demonstrates that many useful points of a systematic goal tree analysis can be incorporated into forensic investigations. These include:

- Reuse of knowledge about previous similar cases, shown by the common upper branches of the goal tree.
- Formulation and execution of an investigation strategy and advance planning to overcome known obstacles, such as analyzing copies of the contract and email messages rather than the originals.
- Formulation and analysis of alternative hypotheses, such as if the anomalies in the time zones in email headers indicate fraud or have alternative explanations.
- Clarification of the reliance on assumptions. The opposing parties agreed that only one contract was signed by Zuckerberg, an assumption needed to prove that the work for hire contract is a forgery after showing that the StreetFax contract is authentic.
- Explanation of the overall argument in the case by combining all the claims in each branch into a coherent, comprehensive and consistent narrative.

One limitation is the absence of a detailed analysis of timelines and timestamps that is crucial to most investigations. The goal tree de-

composition may suggest possible avenues for investigation by creating requirements to discover anomalous temporal metadata, but they would be broad and possibly difficult for an analyst to perform. We plan to investigate how the goal tree analysis may inform and integrate with a timeline tool.

6. Conclusions

The systematic digital forensic investigation process presented in this paper has four main activities for understanding the context of incidents. The activities include the identification and analysis of the goals, the identification and analysis of obstacles, the identification and execution of the required actions, and the operations that must be applied to satisfy the main investigation goals. The application to the real-world contract forgery case of Ceglia v. Zuckerberg and Facebook demonstrates that the process can effectively capture the various forensic and anti-forensic aspects of an investigation.

Our future research will focus on defining a framework for the extraction of common patterns for describing goal-driven digital forensic investigations along with their obstacles and operationalization. Document forgery as in Ceglia v. Zuckerberg and Facebook would be an excellent domain to investigate. A limitation of the paper is the use of an existing case study, which is overcome by comprehensively modeling the entire case. To address this limitation, we plan to construct a general model that could be applied to document forgery cases. Additionally, we plan to utilize formal languages and formal verification tools to provide more rigor in specifying forensic investigations and to prove claims with a high level of assurance.

References

[1] B. Aziz, Towards requirements-driven digital forensic investigations, *Proceedings of the Second International Conference on Cyber Crime, Security and Digital Forensics*, 2012.

[2] H. Blodget, The guy who says he owns 50% of Facebook just filed a boatload of new evidence – and it's breathtaking, *Business Insider*, April 12, 2011.

[3] N. Broom, Declaration of Neil Broom, Paul D. Ceglia v. Mark Elliot Zuckerberg and Facebook, Inc., United States District Court, Western District of New York, Civil Action 1:10-cv-569-RJA-LGF, Technical Resource Center, Atlanta, Georgia, 2012.

[4] B. Carrier and E. Spafford, An event-based digital forensic investigation framework, *Proceedings of the Digital Forensics Research Workshop*, 2004.

[5] E. Casey and C. Rose, Forensic analysis, in *Handbook of Digital Forensics and Investigation*, E. Casey (Ed.), Elsevier Academic Press, Burlington, Massachusetts, pp. 21–62, 2010.

[6] K. Dahbur and B. Mohammad, The anti-forensics challenge, *Proceedings of the International Conference on Intelligent Semantic Web-Services and Applications*, article no. 14, 2011.

[7] A. Fuxman, R. Kazhamiakin, M. Pistore and M. Roveri, Formal Tropos: Language and Semantics, Technical Report, Department of Information and Communication Technology, University of Trento, Trento, Italy (`disi.unitn.it/~ft/papers/ftsem03.pdf`), 2003.

[8] R. Harris, Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem, *Digital Investigation*, vol. 3(S), pp. S44–S49, 2006.

[9] M. Huber, M. Mulazzani, M. Leithner, S. Schrittwieser, G. Wondracek and E. Weippl, Social snapshots: Digital forensics for online social networks, *Proceedings of the Twenty-Seventh Annual Computer Security Applications Conference*, pp. 113–122, 2011.

[10] P. Hunton, The growing phenomenon of crime and the Internet: A cyber crime execution and analysis model, *Computer Law and Security Review*, vol. 25(6), pp. 528–535, 2009.

[11] D. Kahvedzic and T. Kechadi, DIALOG: A framework for modeling, analysis and reuse of digital forensic knowledge, *Digital Investigation*, vol. 6(S), pp. S23–S33, 2009.

[12] G. LaPorte, Paul D. Ceglia v. Mark Elliot Zuckerberg and Facebook, Inc., United States District Court, Western District of New York, Civil Action 1:10-cv-00569-RJA, Document 326, Riley Welch LaPorte and Associates Forensic Laboratories, Lansing, Michigan, 2012.

[13] R. Leigland and A. Krings, A formalization of digital forensics, *International Journal of Digital Evidence*, vol. 3(2), 2004.

[14] S. O'Ciardhuain, An extended model of cyber crime investigations, *International Journal of Digital Evidence*, vol. 3(1), 2004.

[15] G. Palmer, A Road Map for Digital Forensic Research, DFRWS Technical Report DTR-T001-01 Final, Digital Forensic Research Workshop, Utica, New York (`www.dfrws.org/2001/dfrws-rm-final.pdf`), 2001.

[16] M. Reith, C. Carr and G. Gunsch, An examination of digital forensic models, *International Journal of Digital Evidence*, vol. 1(3), 2002.

[17] F. Romano, Report and Recommendation, Paul D. Ceglia v. Mark Elliot Zuckerberg and Facebook, Inc., United States District Court, Western District of New York, Civil Action 1:10-cv-00569-RJA, Document 327, Rochester, New York, 2012.

[18] Stroz Friedberg, Report of Digital Forensic Analysis in: Paul D. Ceglia v. Mark Elliot Zuckerberg, Individually, and Facebook, Inc., United States District Court, Western District of New York, Civil Action 1:10-cv-00569-RJA, New York (`www.wired.com/images_blogs/threatlevel/2012/03/celiginvestigation.pdf`), 2012.

[19] Technical Working Group for Electronic Crime Scene Investigation, Electronic Crime Scene Investigation: A Guide for First Responders, Technical Report, National Institute of Justice, Washington, DC, 2001.

[20] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, Wiley, Chichester, United Kingdom, 2009.

IV

FILESYSTEM FORENSICS

Chapter 12

FILE FRAGMENT ANALYSIS USING NORMALIZED COMPRESSION DISTANCE

Stefan Axelsson, Kamran Ali Bajwa and Mandhapati Venkata Srikanth

Abstract The first step when recovering deleted files using file carving is to identify the file type of a block, also called file fragment analysis. Several researchers have demonstrated the applicability of Kolmogorov complexity methods such as the normalized compression distance (NCD) to this problem. NCD methods compare the results of compressing a pair of data blocks with the compressed concatenation of the pair. One parameter that is required is the compression algorithm to be used. Prior research has identified the NCD compressor properties that yield good performance. However, no studies have focused on its applicability to file fragment analysis. This paper describes the results of experiments on a large corpus of files and file types with different block lengths. The experimental results demonstrate that, in the case of file fragment analysis, compressors with the desired properties do not perform statistically better than compressors with less computational complexity.

Keywords: File fragment analysis, file carving, normalized compression distance

1. Introduction

During the course of an examination, a digital forensic practitioner is often faced with a collection of file fragments from slack space on disks, USB flash drives, etc. Sometimes enough metadata (e.g., file links and headers) survives to make reconstruction easy, but often, what exists is a random collection of file fragments that have to be put together to form partially-complete files. Knowing or having an indication of the types of the file fragments (e.g., pictures, executables or text) aids in the reconstruction process. Knowing the file type of the blocks reduces

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 171–182, 2013.

the number of block combinations that must be attempted to only those corresponding to the specific file type.

Several techniques can be used to identify the file type of a block. This paper focuses on the use of a learning algorithm based on the normalized compression distance (NCD) [9]. NCD is based on the idea that if two compressed data samples are close to the compressed combination of the two samples, then they are more similar than two data samples that, after compression, are far from the compressed combination.

Previous research has applied NCD to file fragment analysis [3, 18]. When developing an NCD file fragment classifier, it is necessary to select the compressor that yields the best performance. This paper investigates which compression performance metrics have the greatest impact on file type classification performance.

2. Related Work

The identification of file types from file fragments is an active field of research [1, 2, 14, 16, 17]. While many different methods have been developed, this section focuses on methods that use machine learning algorithms. It should be noted that the NCD approach performs on par with the best methods for some file types.

The most relevant work is by Veenman [18], who focused on the n-valued problem ($n = 11$) in a 450 MB data set using statistical methods (including a method based on Kolmogorov complexity as is NCD). Veenman reports an overall accuracy of 45%, which is similar to the results of Axelsson, *et al.* [3, 4]. Veenman also developed a set of two-class (cross-type) classifiers that exhibit higher classification accuracy. Calhoun, *et.al.* [6] extended the work of Veenman, but limited their research to four file types that were analyzed in pairs; this complicates the direct comparison of the results obtained by the two efforts.

Cebrian, *et al.* [8] have attempted to identify the most appropriate compressor for NCD. In particular, they evaluated the `gzip`, `bzip2` and `ppmd` compressors and found that the best compressor choice was the one with the highest compression ratio and strongest idempotency. However, the experimental results presented in this paper do not match this finding for the file fragment analysis problem. One hypothesis for the difference is that the input data of Cebrian, *et al.* is several orders of magnitude larger, and their focus is on when the input becomes too large for a compressor to handle (i.e., when the concatenated input is larger than the window size of the compressor, if one exists). Our work investigates the other end of the spectrum where the input is very small; thus, the results obtained are quite different.

3. Normalized Compression Distance

NCD is based on the idea that using a compression algorithm on data vectors both individually and concatenated provides a better measure of the distance between the two vectors. The better the combined vectors compress compared with how the individual vectors compress (normalized to remove differences in length between the sets of vectors), the more similar are the two vectors. The NCD metric is given by:

$$NCD(x,y) = \frac{C(x,y) - min(C(x), C(y))}{max(C(x), C(y))}$$

where $C(x)$ is the compressed length of x, and $C(x,y)$ is the compressed length of x concatenated with y.

In order to apply this metric in supervised learning, it is necessary to select features of the input data used for training. The distances from the features of the training instances to the features of the instances being classified are then computed.

The k-nearest neighbor (kNN) algorithm is commonly used with NCD for classification. In kNN, the k nearest feature vectors are selected and the class of the new sample is assigned according to a majority vote. For example, with $k = 10$, if three of the closest feature vectors are of file type `zip` and seven are `exe`, then the feature vector being classified is assigned to the class `exe` although the closest example might have been a `zip` feature vector.

An advantage of NCD is that anything that can be put through the compression algorithm can be used as a vector [8]. This means that NCD is parameter free and resistant to errors in feature selection. Not having to make any feature selection is a substantial advantage as there are almost an infinite number of ways to select and evaluate features.

4. Research Questions

In order to develop an NCD-based file fragment classifier, a central question is which compression algorithm to use. The determination of the best compressor is best done by evaluating how the candidate compressors perform the task of interest.

Previous research has shown that a compressor that performs well in terms of compression ratio and idempotency performs well for NCD when it comes to other data classification tasks [7, 8, 13]. In the case of compression ratio, a compressor that produces a smaller output is the better choice for NCD classification.

An idempotent operation produces the same result no matter how many times it is applied (barring the first application). In the case of

compression for NCD, the following identities should hold:

$$C(xx) \approx C(x) \quad \text{and} \quad C(\lambda) = 0.$$

The first identity specifies that a desirable compressor can detect that input is repeated and store very little extra information to remember this fact. The second specifies that the compression of an empty input yields an output of length zero. The two identities are unattainable in practice, but they serve as useful benchmarks.

We evaluate three compressors: (i) `gzip` (using the DEFLATE algorithm) [11, 12]; (ii) `bzip2` (using the Burrows-Wheeler transform) [5]; and (iii) `ppmd` (a version of the PPM algorithm) [10]. The three compressors were selected because of their popularity and availability, including open source implementations and algorithm descriptions. Also, they demonstrate the progressive advancement in compression schemes. In fact, both the compression ratio and resource consumption for the three compressors have the following relationship [8]:

$$\texttt{gzip} < \texttt{bzip2} < \texttt{ppmd}.$$

This research evaluates the combined effect of compression ratio and idempotency when used for file fragment analysis on several different block sizes. The reason for considering several block sizes is that different block sizes are encountered in the field, and that it is conceivable that fragment size could affect compressor performance. By exposing the compressor to more data, it could conceivably see more of the file structure, and be better able to classify the fragment.

More formally, the hypotheses posed are:

- **Null Hypothesis (H0):** There is no difference between the idempotencies measured by the compressors.
  Alternative Hypothesis (H1): There is a difference between the idempotencies measured by the compressors.

- **Null Hypothesis (H2):** There is no effect on the measured idempotencies of the compressors by changing block size.
  Alternative Hypothesis (H3): There is an effect on the measured idempotencies of the compressors by changing block size.

Next, we evaluate how the classification performance is affected. We consider the following hypotheses:

- **Null Hypothesis (H0):** There is no difference in the NCD classification performance when a compressor is selected on the basis of compression ratio or idempotency.

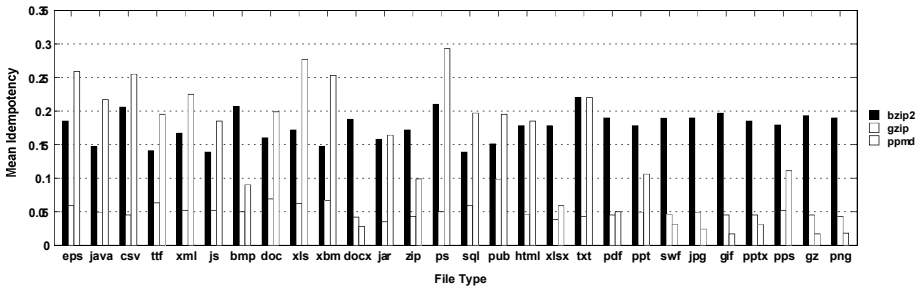


Figure 1. Mean idempotency for all files and all block sizes.

Alternative Hypothesis (H1): There is a difference in the NCD classification performance when a compressor is selected on the basis of compression ratio or idempotency.

- **Null Hypothesis (H2):** There is no difference in the NCD classification performance when the block size is changed.
  Alternative Hypothesis (H3): There is a difference in the NCD classification performance when the block size is changed.

5. Experiments

The testing data contained selected blocks from files in the Garfinkel corpus [15]. The data comprised 50 files of each of the following 28 types: `pdf`, `html`, `jpg`, `txt`, `doc`, `xls`, `ppt`, `xms`, `gif`, `ps`, `csv`, `gz`, `eps`, `png`, `swf`, `pps`, `sql`, `java`, `pptx`, `docx`, `ttf`, `js`, `pub`, `bmp`, `xbm`, `xlsx`, `jar` and `zip`, yielding a total of 1,400 files.

For testing, the files were broken down into block sizes of 512, 1,024, 1,536 and 2,048 bytes. The idempotency of each block was calculated as:

$$NCD(x,x) = \frac{C(x,x) - min(C(x), C(x))}{max(C(x), C(x))} = \frac{C(x,x) - C(x)}{C(x)}.$$

An average for all the blocks of a given file and the average idempotency for all the files of a given file type were calculated for all the block sizes. Figures 1 and 2 show the means and standard deviations of the idempotencies for each of the file types.

Figure 1 shows that `gzip` has better (lower) idempotency values than `bzip2`, which has better values than `ppmd`. Analysis of the standard deviation of the means in Figure 2 reveals that the same pattern appears to hold: `gzip` is more consistent in idempotency while `bzip2` is worse, and `ppmd` is similar to `bzip2`, although it varies wildly for some file types.

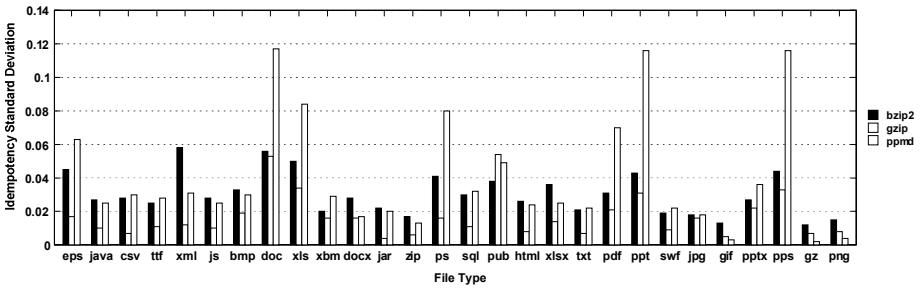


Figure 2. Standard deviations of idempotencies for all files and all block sizes.

We have no scientific explanation for these results. However, one hypothesis is that the fixed overhead plays a part in the compression of what is considered to be a short input by compression algorithm standards.

Table 1. ANOVA test for the null hypothesis.

Source of Variation	SS	df	MS	F	P-Value	F-Crit
Between Groups	0.317	2	0.158	63.46	2.62^{-17}	3.109
Within Groups	0.202	81	0.00249			
Total	0.519	83				

Table 1 lists the results of an ANOVA test on the null hypothesis that there is no difference in idempotencies for the compressors. The null hypothesis is rejected because F is much greater than F-Critical and the P-Value is less than 0.05. Thus, the differences in idempotency for this type of data and for these compressors are statistically significant.

Testing the second null hypothesis that there is no effect on the measured idempotencies when changing the block size involves three cases:

- **Case 1: H0:** Block size original = Block size changed.
  Alt. Hypot. H1: Block size original $\neq$ Block size changed.
- **Case 2: H2:** `bzip2` = `gzip` = `ppmd`.
  Alt. Hypot. H3: `bzip2` $\neq$ `gzip` $\neq$ `ppmd`.
- **Case 3: H4:** All interaction is absent.
  Alt. Hypot. H5: An interaction is present.

Table 2 ("Sample" row) confirms that for Case 1, H0 is valid and that changing the block size has no significant effect on the idempotencies of the three compressors. The "Columns" row shows that for Case 2, H2 is rejected because there is no difference between the idempotencies

Table 2. ANOVA test for the multi-case hypothesis.

Source of Variation	SS	df	MS	F	P-Value	F-Crit
Sample	0.0024	3	0.0008	0.232	0.873	2.63
Columns	0.936	2	0.468	135.388	1.83^{-43}	3.023
Interaction	0.066	6	0.011	3.203	0.0045	2.126
Within Groups	1.12	324	0.00346			
Total	2.125	335				

measured by the three compressors. For Case 3, the "Interaction" row shows that F is greater than F-Critical, but the P-Value is greater than 0.05. Thus, there is tentative support to reject H4, but the result is not significant. In other words, while the experiment measures a difference, the difference might be due to chance or some other factors.

These results are interesting and not obvious. There is clearly a difference in the measured idempotencies for the three compressors with `gzip` having a smaller idempotency value than `bzip2`, whose value is again smaller than the value for `ppmd`. The better the compressor in the general case, the worse its idempotency for file fragment analysis. Furthermore, the idempotency is unaffected by block size, which is less surprising given that 2,048 bytes is a small amount of data compared with the amount of data that these algorithms are designed to handle.

The second set of tests focused on the classification performance of NCD-based file fragment classification given that there are differences in compression performance and idempotency between the compressors. The second set evaluated the `bzip2` and `gzip` compressors on a different selection of file fragments from the same corpus. The `ppmd` compressor was excluded because of its lengthy computational time and because it has a similar compression ratio and idempotency as `bzip2`.

We formulate the following experimental hypotheses:

- **Null Hypothesis (H0):** Although the compressors have different compression ratios and idempotencies, there is no difference in classification performance.
  Alternative Hypothesis (H1): There is a difference in classification performance.

- **Null Hypothesis (H2):** There is no difference in classification performance when the block size is changed.
  Alternative Hypothesis (H3): There is a difference in classification performance.

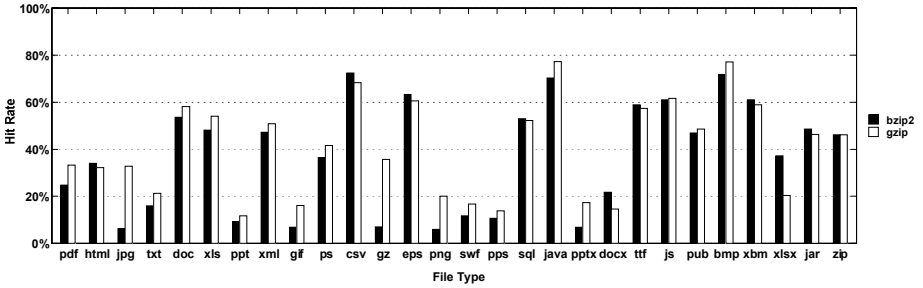


Figure 3. Hit rates for a block size of 512 bytes (average over all k values).

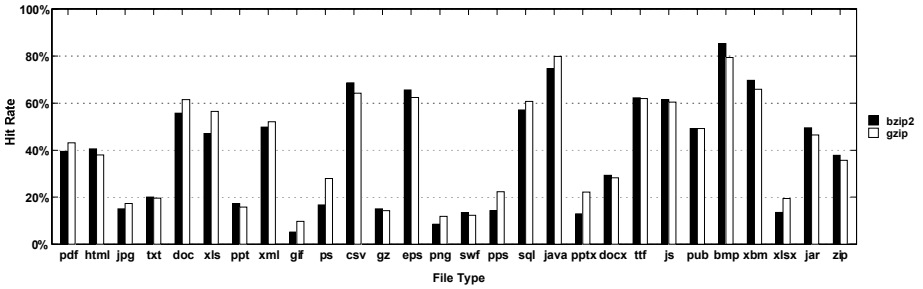


Figure 4. Hit rates for a block size of 1,024 bytes (average over all k values).

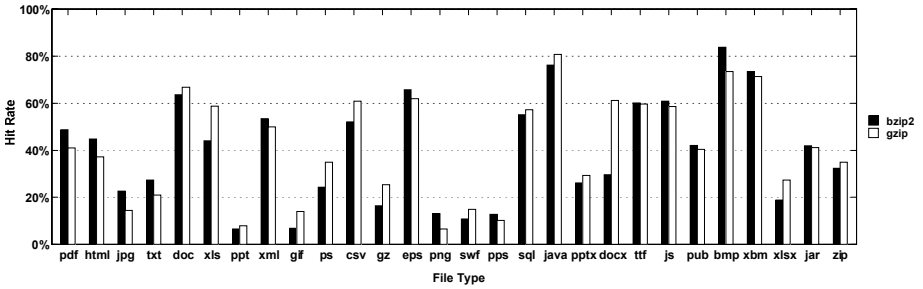


Figure 5. Hit rates for a block size of 1,536 bytes (average over all k values).

This experiment followed the methodology discussed in [3]. The data was divided into ten slices and a form of ten-fold cross validation was performed. kNN classification was then run for each of the ten blocks.

Figures 3, 4, 5 and 6 show the hit rates for various block sizes averaged over all the k values (1 through 10). The figures show similar results for both compressors. While one compressor outperforms the other by a small margin for some file types, the other does the same for other types.

Figures 7 and 8 do not show any clear trends regarding block sizes. The results are similar, although there are differences for certain combinations of block size (e.g., `xlsx` for `bzip2`, but not for `gzip`).

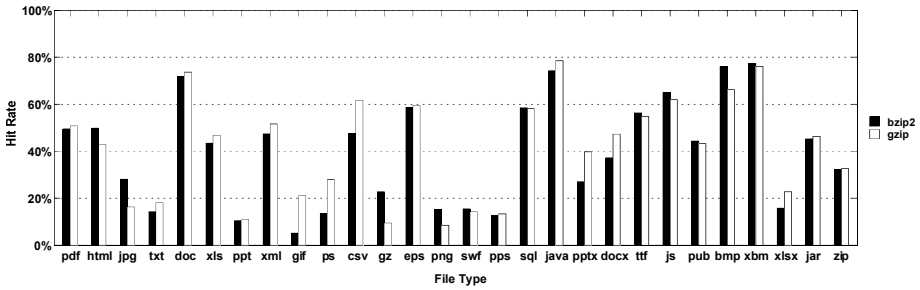


Figure 6. Hit rates for a block size of 2,048 bytes (average over all k values).

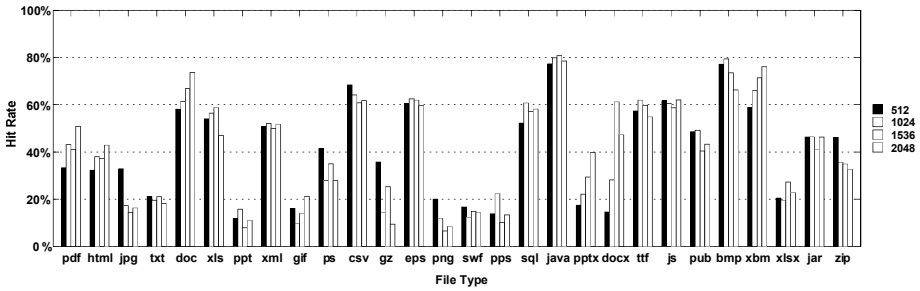


Figure 7. Hit rates for all block sizes (average over all k values, `gzip`).

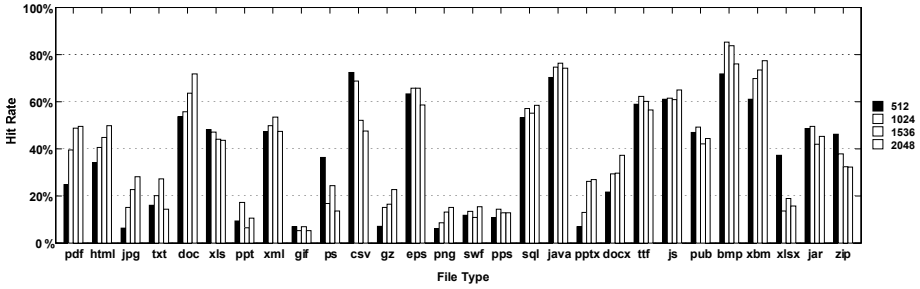


Figure 8. Hit rates for all block sizes (average over all k values, `bzip2`).

An ANOVA test was applied to each hypothesis to test whether or not choosing a compression algorithm based on established measures of effectiveness leads to a better classification outcome. Table 3 shows that there is no statistically significant difference between the classification accuracy of the two classifiers.

The second hypothesis involves multiple cases:

- **Case 1: H0:** Block size original = Block size changed.
  Alt. Hypot. H1: Block size original $\neq$ Block size changed.

- **Case 2: H2:** `bzip2` = `gzip`.
  Alt. Hypot. H3: `bzip2` $\neq$ `gzip`.

Table 3. ANOVA test of difference between compressors regarding accuracy.

Source of Variation	SS	df	MS	F	P-Value	F-Crit
Between Groups	2,794.876	19	147.1	0.287	0.998	1.606
Within Groups	277,059.8	540	513.07			
Total	2.125	335				

- **Case 3: H4:** All interaction is absent.
  Alt. Hypot. H5: An interaction is present.

Table 4. ANOVA test of difference between compressors regarding block size.

Source of Variation	SS	df	MS	F	P-Value	F-Crit
Sample	972.3	3	324.1	0.63	0.597	2.609
Columns	6,653.9	19	350.2	0.678	0.844	1.59
Interaction	874.7	57	15.35	0.029	1	1.33
Within Groups	1,115,723	2,160	516.5			
Total	1,124,224	2239				

The corresponding ANOVA test results are shown in Table 4. For Case 1, H0 is retained. The "Sample" row shows that F is less than F-Critical and the P-Value is greater than 0.05. This supports the null hypothesis that there is no significant difference in NCD classification performance of `gzip` and `bzip2` when the block size is changed. For Case 2, H2 is retained because F is less than F-Critical and the P-Value is greater than 0.05. Thus, there is once again no significant difference in the performance when selecting `gzip` or `bzip2`. As in the previous two cases, the null hypothesis H4 is retained for Case 3 – there is no interaction between the two variables.

6. Conclusions

The experimental results demonstrate that there is no significant effect on file fragment analysis accuracy for the NCD algorithm when using different block sizes. Also, there is no significant effect on performance when a compressor is selected on the basis of better compression ratio or idempotency.

Much research remains to be done concerning files and file types (e.g., how to handle a `png` image stored as part of a Word document), especially for the difficult, high entropy files such as already compressed input and encrypted files. Also, since the performance measures of com-

pression ratio and idempotency do not appear to be useful in selecting a compression algorithm, future research should focus on creating new performance measures geared specifically for NCD-based file fragment analysis.

References

[1] I. Ahmed, K. Lhee, H. Shin and M. Hong, On improving the accuracy and performance of content-based file type identification, *Proceedings of the Fourteenth Australasian Conference on Information Security and Privacy*, pp. 44–59, 2009.

[2] L. Aronson and J. van den Bos, Towards an engineering approach to file carver construction, *Proceedings of the Thirty-Fifth Annual IEEE Computer Software and Applications Conference Workshops*, pp. 368–373, 2011.

[3] S. Axelsson, The normalized compression distance as a file fragment classifier, *Digital Investigation*, vol. 7(S), pp. S24–S31, 2010.

[4] S. Axelsson, Using normalized compression distance for classifying file fragments, *Proceedings of the International Conference on Availibility, Reliability and Security*, pp. 641–646, 2010.

[5] M. Burrows and D. Wheeler, A Block-Sorting Lossless Data Compression Algorithm, Technical Report No. SRC-RR-124, Digital Equipment Corporation Systems Research Center, Palo Alto, California, 1994.

[6] W. Calhoun and D. Coles, Predicting the types of file fragments, *Digital Investigation*, vol. 5(S), pp. S14–S20, 2008.

[7] M. Cebrian, M. Alfonseca and A. Ortega, Common pitfalls using normalized compression distance: What to watch out for in a compressor, *Communications in Information and Systems*, vol. 5(4), pp. 367–400, 2005.

[8] M. Cebrian, M. Alfonseca and A. Ortega, The normalized compression distance is resistant to noise, *IEEE Transactions on Information Theory*, vol. 53(5), pp. 1895–1990, 2007.

[9] R. Cilibrasi, Statistical Inference Through Data Compression, Ph.D. Thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands, 2007.

[10] J. Cleary and I. Witten, Data compression using adaptive coding and partial string matching, *IEEE Transactions on Communications*, vol. 32(4), pp. 396–402, 1984.

[11] P. Deutsch, DEFLATE Compressed Data Format Specification Version 1.3, RFC 1951, 1996.

[12] P. Deutsch, GZIP File Format Specification Version 4.3, RFC 1952, 1996.

[13] R. Feldt, R. Torkar, T. Gorschek and W. Afzal, Searching for cognitively diverse tests: Towards universal test diversity metrics, *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation*, pp. 178–186, 2008.

[14] S. Fitzgerald, G. Mathews, C. Morris and O. Zhulyn, Using NLP techniques for file fragment classification, *Digital Investigation*, vol. 9(S), pp. S44–S49, 2012.

[15] S. Garfinkel, P. Farrell, V. Roussev and D. Dinolt, Bringing science to digital forensics with standardized forensic corpora, *Digital Investigation*, vol. 6(S), pp. S2–S11, 2009.

[16] Q. Li, A. Ong, P. Suganthan and V. Thing, A novel support vector machine approach to high entropy data fragment classification, *Proceedings of the South African Information Security Multi-Conference*, pp. 236–247, 2010.

[17] W. Li, K. Wang, S. Stolfo and B. Herzog, Fileprints: Identifying file types by n-gram analysis, *Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop*, pp. 64–71, 2005.

[18] C. Veenman, Statistical disk cluster classification for file carving, *Proceedings of the Third International Symposium on Information Assurance and Security*, pp. 393–398, 2007.

Chapter 13

QUANTIFYING WINDOWS FILE SLACK SIZE AND STABILITY

Martin Mulazzani, Sebastian Neuner, Peter Kieseberg, Markus Huber, Sebastian Schrittwieser and Edgar Weippl

Abstract Slack space can be used to hide data from the operating system and other users. While some forms of data hiding are easily detectable, others are subtle and require an experienced forensic practitioner to discover the hidden data. The amount of data that can be hidden varies with the type of slack space and environmental parameters such as filesystem block size and partition alignment. This paper evaluates the amount of file slack space available in Windows systems and the stability of slack space over time with respect to system updates. Measurements of the file slack for eighteen versions of Microsoft Windows with the NTFS filesystem reveal that many of the files change very little during system updates and are, thus, highly suitable for hiding data. A model is presented for estimating the amount of data that can be hidden in the file slack space of Windows filesystems of arbitrary size.

Keywords: Forensic analysis, file slack space, Windows systems, data hiding

1. Introduction

With ever increasing hard drive and memory storage capacities, the examination of slack space has become an important component of digital forensic investigations. Hard drives with 3 TB or more capacity are commonly available, which leaves plenty of space to hide data and make manual disk forensic analysis cumbersome and time-consuming. While encryption can be used to make hard drive data inaccessible during a forensic examination [4], slack space data hiding and steganography [8] conceal data more or less in plain sight.

Data hidden in certain areas of a hard drive, such as the host protected area (HPA) and device configuration overlay (DCO), can be detected by current tools. However, it is more difficult to find data that was

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 183–193, 2013.

intentionally hidden in file slack space. During the natural life cycle of files on a hard drive, file slack is constantly created and overwritten, and is, thus, not necessarily stable. Moreover, several tools have been developed to hide data – including encrypted data – in file slack space.

To help address the problem of data hiding in slack space, this paper quantifies the file slack space for Microsoft Windows operating systems and how the slack space is affected by system updates. The analysis uses eighteen versions of Microsoft Windows, including Windows XP and Server 2003 through Windows 8 and Server 2012 RC. Based on the analysis, a model is presented for estimating the amount of file slack space across all files on a hard drive.

2. Background

File slack space is a byproduct of operating system filesystem design. To reduce addressing overhead in a filesystem, multiple sectors of a hard drive are clustered together; this implicitly creates "slack space" for every file whose size does not match the cluster size exactly. File slack is defined as the space between the end of the file and the end of the allocated cluster [3]. The size of usable slack space depends on the cluster size of a filesystem, the sector size of the hard drive and the padding strategy used by the operating system. Older hard drives commonly have a sector size of 512 bytes while newer hard drives have a sector size of 4 KiB. FAT32 usually has a cluster size between 4 KiB and 32 KiB depending on the partition size, while NTFS usually has a cluster size of 4 KiB for drives with less than 16 TB capacity [11]. Windows uses padding only for the last sector at the end of each file. The other sectors in the cluster are left untouched [3], leaving up to $n-1$ sectors untouched if the portion of the file in the last cluster (with n sectors) is smaller than the sector size.

Other forms of slack space (e.g., volume slack and partition slack) and data hiding locations (e.g., HPA and DCO) are also encountered in forensic examinations [2, 7]. The benefit of file slack, however, is that it exists on every hard drive that uses a filesystem that clusters sectors. Also, the file slack size can be extended arbitrarily by storing a large number of small files and/or using large cluster sizes (NTFS, for example, supports a cluster size of up to 64 KiB). File slack is also one of the reasons why bitwise copies have to be created during image acquisition [9]. Otherwise, the data stored in file slack space could be lost.

During a forensic examination, file slack space is of interest in two particular cases. The first is when secure deletion software such as

Table 1. Evaluated operating systems.

Operating System	Upd.	SPs	Operating System	Upd.	SPs
Windows XP Pro.	+189	+2	Windows 7 Pro. SP1	+106	—
Windows XP Pro. SP2	+164	+1	Windows 7 Ent.	+212	+1
Windows XP Pro. SP3	+177	—	Windows 7 Ent. SP1	+167	—
Vista Business	+246	+2	Windows 8 RC	+11	—
Vista Business SP1	+72	+1	Server 2003 R2 Std. SP2	+163	—
Vista Business SP2	+143	—	Server 2003 R2 Ent. SP2	+167	—
Vista Ent. SP1	+207	+1	Server 2008 R2 Std.	+148	+1
Vista Ent. SP2	+143	—	Server 2008 R2 Std. SP1	+103	—
Windows 7 Pro.	+156	+1	Server 2012 RC	+6	—

`shred` or WipeFile has been used to securely delete files, but the slack space may still contain fragments of previous data. In the second case, data has been intentionally hidden in slack space using freeware such as `slacker.exe` for NTFS or `bmap` for Linux.

3. Quantifying OS File Slack Space

SleuthKit's `fiwalk` [5] tool was used to analyze file slack in NTFS for eighteen versions of Microsoft Windows. Each system used the NTFS default cluster size of 4 KiB with the underlying 512 byte sector size, and was installed using the default settings. Each installed system was then patched with the available system updates and service packs, including optional updates such as the .NET framework and additional language packs. After each change, `fiwalk` was executed on the acquired disk images. Table 1 lists the complete set of operating systems and the total numbers of system updates and service packs that were installed during the evaluation.

Scripts were used to parse the `fiwalk` XML outputs and calculate the slack space for each disk image. The calculations omitted NTFS filesystem metadata objects (e.g., $MFT and $Bitmap) and files less than a full cluster in size to avoid files that were possibly resident in the NTFS master file table ($MFT) and, thus, were not directly usable as file slack. The file slack persistence calculation compared the file SHA-1 hash values and timestamps of the last write accesses before and after the updates, assuming that modifying a file could destroy file slack (by increasing or decreasing the file size above the sector boundaries in the last cluster). This was done to prevent overestimation and provide conservative measurements. The data set is available at `www.sba-research.org/team/researchers/martin-mulazzani`.

4. Experimental Evaluation

This section quantifies the file slack available in Windows systems and analyzes the stability of file slack space during system updates.

4.1 Quantification of Slack Space

The amount of available file slack space depends on the number of files and the complexity of the underlying operating system. The experimental results show that more recent operating systems have a larger file base and, therefore, a larger quantity of file slack space. The base installation of Windows XP, the oldest system in the evaluation, can be used to hide about 22 MB of data in file slack while the newest versions of Windows, Windows 8 and Server 2012 RC, can hide 86 MB and 70 MB, respectively. Windows Vista (Business and Enterprise versions) allows approximately 62 MB in file slack while Windows 7 (Professional and Enterprise) allows a little more than 69 MB. A similar trend is observed for Windows server: Windows Server 2003 R2 has about 20 MB of file slack capacity while Server 2008 R2 has about 73 MB of capacity.

One of the key observations is that the amount of file slack increases with system updates, especially with service packs. This is because old system files are retained in the event that something goes wrong during the update process. Table 1 shows the number of system updates that were installed. Many system updates affected only a small number of files, while service packs, which are major updates, can affect many files. An increase in file slack capacity of more than 100% during the lifetime of an operating system is not uncommon; on the average, the slack space doubles in capacity. At the end of the evaluation process, Windows XP had more than 30 MB, Vista 105 MB and Windows 7 Professional 100 MB. Windows 7 Enterprise showed an exceptional increase of more than 500%, from around 72 MB to more than 400 MB. Windows 8 and Server 2012 RC were stable as there were few updates available. Table 2 presents the detailed results.

Figure 1 presents a box plot of the cumulative slack space over all the collected states in the evaluation grouped by product line. This graph can be used by forensic practitioners to assess the amount of possible file slack for a given operating system. Depending on the install date of the operating system, 100 MB to 200 MB of file slack space is not uncommon for newer operating systems such as Windows 7 and Server 2008, even with the default cluster size of 4 KiB in NTFS. The numbers of samples (respectively update steps) for the different product lines were: 15 for Windows XP, 32 for Vista, 19 for Windows 7, and four for 8 RC. In the

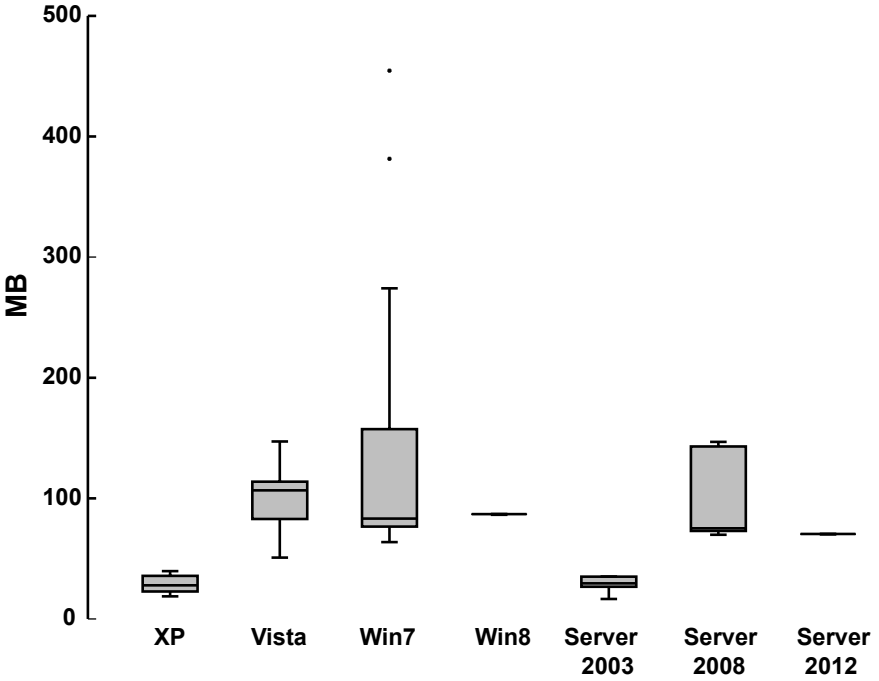


Figure 1. Box plot of quantified file slack space.

case of the Windows servers, Server 2003 R2 had ten, Server 2008 R2 had nine and Server 2012 RC had four.

4.2 Stability of Slack Space

The file slack stability evaluation compared the SHA-1 hash values and timestamps for each file in the images before and after updates. While the numbers of files increased substantially (especially with service packs), the stability of the initial file slack was high – in some cases, more than 90% of the slack space from the initial install was not modified. Also, while the older versions of Windows XP and Server 2003 had 20 MB or less that was persistent with respect to all available system updates, Vista, Windows 7 and Server 2008 had 50 MB or more slack space that was persistent. On the average and across the different Windows versions, 44 MB and 78% of the initial file slack were available at the end of the evaluation process. This means that up to two-thirds of the file slack created during installation was not modified by the end of the analysis. For Server 2003 R2, up to 80% was available. Vista SP2 from 2009 and Windows 7 SP1 from 2011 had more than 90% of their files intact. Further data from real systems in use is needed to confirm these

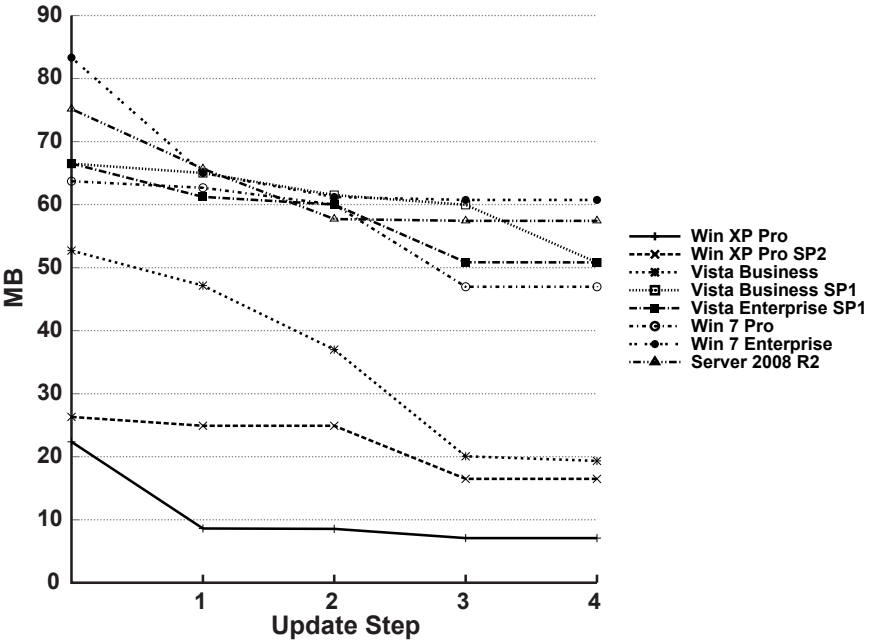


Figure 2. Stability of file slack space across update steps.

results. Figure 2 shows the file slack stability for a selection of Windows versions and the corresponding numbers of update steps. Table 2 shows the detailed results.

Note that, although the total file slack capacity increases with system updates and update steps, the stability of file slack from the initial installation is represented as a monotonically decreasing function. This is because updates can change existing files without adding new files.

Table 3 shows the top file extensions and the most stable file types with respect to slack space for three Windows versions. The number per file extension is the number of files of the type that persisted through all system updates, keeping their slack space untouched.

The most stable files were program files (`.dll`, `.exe`, `.sys`); pictures (`.png`, `.jpg`); and text files (`.chm`, `.htm`). For the three representative images, the first operating system, Windows XP Pro SP2, had a total of 11,723 persistent files, Windows 7 Pro had 29,561 and Server 2008 R2 had 36,394. The slack space per file type was not calculated, but the formula for slack space capacity estimation, presented in the next section, can be used to compute this value.

Table 2. Detailed results of file slack quantification.

Operating System	Initial Slack	Final Slack	Stability
Windows XP Pro.	22.36 MB	36.97 MB (165%)	7.09 MB/31.7%
Windows XP Pro. SP2	26.31 MB	29.49 MB (112%)	16.49 MB/62.7%
Windows XP Pro. SP3	18.72 MB	23.15 MB (124%)	14.21 MB/75.9%
Vista Business	52.70 MB	147.13 MB (279%)	19.34 MB/36.7%
Vista Business SP1	66.49 MB	119.89 MB (180%)	50.77 MB/76.4%
Vista Business SP2	50.89 MB	82.99 MB (163%)	48.13 MB/94.7%
Vista Ent. SP1	66.51 MB	140.35 MB (211%)	50.82 MB/76.4%
Vista Ent. SP2	71.73 MB	113.76 MB (159%)	67.36 MB/93.9%
Windows 7 Pro.	63.71 MB	115.16 MB (181%)	46.96 MB/73.7%
Windows 7 Pro. SP1	65.03 MB	77.80 MB (120%)	60.73 MB/93.4%
Windows 7 Ent.	83.33 MB	454.62 MB (546%)	60.74 MB/72.9%
Windows 7 Ent. SP1	65.10 MB	381.56 MB (586%)	60.77 MB/93.3%
Windows 8 RC	86.40 MB	87.06 MB (101%)	65.10 MB/75.3%
Server 2003 R2 Std. SP2	24.42 MB	33.90 MB (140%)	20.13 MB/82.4%
Server 2003 R2 Ent. SP2	16.55 MB	35.13 MB (212%)	15.20 MB/91.8%
Server 2008 R2 Std.	75.16 MB	146.80 MB (195%)	57.43 MB/76.4%
Server 2008 R2 Std. SP1	69.82 MB	73.03 MB (105%)	69.19 MB/99.1%
Server 2012 RC	70.16 MB	70.58 MB (101%)	70.01 MB/99.8%

Table 3. Top file types with respect to slack space stability.

XP Pro SP2		Win7 Pro		Server 2008 R2	
`.dll`	4,414	`.dll`	6,302	`.dll`	7,303
`.exe`	1,106	`.mui`	3,906	`.cat`	6,752
`.sys`	793	`.est`	3,190	`.est`	4,204
`.inf`	692	`.inf`	1,352	`.mui`	3,907
`.pnf`	674	`.gpd`	1,303	`.exe`	1,364
`.chm`	317	`.exe`	1,067	`.gpd`	1,303
`.htm`	233	`.png`	1,051	`.inf`	1,160
`.nls`	192	`.cat`	945	`.mum`	909
`.jpg`	168	`.pnf`	914	`.ppd`	806
Total	8,607	Total	20,030	Total	27,708

4.3 Discussion

While the earliest versions of Windows XP had little more than 10,000 files that were larger then the default cluster size of 4 KiB, Windows 7 had more than 40,000 such files and Windows 8 had more than 55,000. The number of files increases significantly with service packs: Windows Vista, which started with 35,000 files, had more than 90,000 files after installing two service packs. For forensic practitioners, this is particu-

larly important because it appears that a considerable amount of slack space is available to hide data.

An adversary that actively uses file slack could further increase the cluster size of NTFS – up to 64 KiB is supported by NTFS. A quick estimate obtained by running our scripts on an `.xml` file chosen at random showed that increasing the cluster size to 32 KiB on Windows Vista with more than 90,000 files could increase the file slack to 1.25 GB. This is more than eight times larger than the 150 MB available for the default cluster size of 4 KiB. Therefore, forensic practitioners should pay close attention to artificially large cluster sizes, especially because they are uncommon and can be instrumented to hide a large amount of data in the file slack.

The approach for measuring file slack has some limitations. First, the tests ignored the fact that modern hard drives have a default sector size of 4 KiB. This was necessary in order to evaluate commonly used operating systems such as Windows XP that do not support hard drives with larger sectors [12]. Second, user activity was excluded because there is no adequate method that simulates user activity in a realistic fashion. Third, additional files that are commonly found in Windows systems due to software installed by users were excluded. Because of these limitations, the results represent a conservative upper bound for operating-system-specific filesystem slack and a lower bound when including user data.

5. File Slack Space Estimation

This section presents a formula that generalizes the evaluation results and provides a means for a digital forensic practitioner to determine the expected size of the slack space.

Let n be the number of files and k be the cluster size (i.e., number of sectors in a cluster). Furthermore, let s denote the number of bytes per sector. Since only the last cluster of a file may not be filled completely (i.e., all the clusters of a file except for one cluster do not provide slack space), the slack space that can be expected from a single file is completely independent of the actual size of the file. Thus, the expected value S_E of the average slack space of a file is given by:

$$S_E = s\mathbb{E}(X)$$

where $\mathbb{E}(X)$ is the expected value with respect to the underlying statistical distribution of the fill-grade of the last cluster.

In general, it can be assumed that the fill-grade of the last cluster is equally distributed among all possible values. Since the first sector inside a cluster is always filled in the case of NTFS, the number of free

sectors that can be used is an element of the set $\{1, \ldots, k-1\}$. This yields:

$$\begin{aligned} S_{(n,s,k)} &= ns\mathbb{E}(X) \\ &= ns\sum_{x=1}^{k-1} x\frac{1}{k} \\ &= ns\frac{1}{k}\frac{(k-1)k}{2} \\ &= ns\frac{k-1}{2}. \end{aligned}$$

Using a typical sector size $s = 512$ bytes and $k = 8$ sectors per cluster, the formula above reduces to:

$$S_n = 1792n.$$

6. Related Work

Recent research related to slack space has examined the use of file fragmentation to hide data in FAT partitions [10]. FragFS [14] uses slack space in $MFT entries to hide data, leveraging the fact that $MFT entries commonly have a fixed length of 1,024 bytes per entry. According to estimates [14], a total of about 36 MB of data can be hidden in $MFT slack of Windows XP systems because each $MFT entry uses less than 450 bytes on the average.

Researchers have studied slack space in cloud environments and in hard drives purchased in the second-hand market. In particular, it is possible to retrieve possibly sensitive data such as deleted SSH keys [1] and to hide data without leaving traces in a cloud client or in the log files of a cloud service provider [13]. Also, Garfinkel and Shelat [6] have shown that slack space is rarely deleted from pre-used hard drives from the second-hand market.

7. Conclusions

Slack space cannot be ignored in digital forensic investigations because it can be used to conceal large amounts of data. The principal contributions of this paper are the quantification of file slack space in eighteen versions of Microsoft Windows, an evaluation of the durability of file slack space under system updates, and a simple model for estimating the total storage capacity of file slack space. All eighteen operating systems that were tested offered initial slack space in the tens of megabytes

and this space was largely immune to system updates. On the average, 44 MB or 78% of the initial file slack was available after installing all the available system updates, including complete service packs. In the case of newer versions of Windows, such as Windows 7 and Server 2008, around 100 MB to 200 MB of slack space was available with the default cluster size of 4 KiB.

Our future work will address the limitations discussed in the paper and will conduct a large-scale survey to measure the slack space in real-world systems. Additionally, it will evaluate the amount of slack space available in user files and their stability. We also plan to study modern operating systems and filesystems that use sector clustering, especially HFS+ in OS X systems and ext4 in Linux systems.

Acknowledgement

This research was funded by the Austrian Research Promotion Agency under COMET K1 and by Grant No. 825747.

References

[1] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda and S. Loureiro, A security analysis of Amazon's Elastic Compute Cloud Service, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Applied Computing*, pp. 1427–1434, 2012.

[2] H. Berghel, Hiding data, forensics and anti-forensics, *Communications of the ACM*, vol. 50(4), pp. 15–20, 2007.

[3] B. Carrier, *File System Forensic Analysis*, Pearson Education, Upper Saddle River, New Jersey, 2005.

[4] E. Casey and G. Stellatos, The impact of full disk encryption on digital forensics, *ACM SIGOPS Operating Systems Review*, vol. 42(3), pp. 93–98, 2008.

[5] S. Garfinkel, Automating disk forensic processing with Sleuthkit, XML and Python, *Proceedings of the Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 73–84, 2009.

[6] S. Garfinkel and A. Shelat, Remembrance of data passed: A study of disk sanitization practices, *IEEE Security and Privacy*, vol. 1(1), pp. 17–27, 2003.

[7] E. Huebner, D. Bem and C. Wee, Data hiding in the NTFS file system, *Digital Investigation*, vol. 3(4), pp. 211–226, 2006.

[8] S. Katzenbeisser and F. Petitolas (Eds.), *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Norwood, Massachusetts, 2000.

[9] K. Kent, S. Chevalier, T. Grance and H. Dang, Guide to Integrating Forensic Techniques into Incident Response, NIST Special Publication 800-86, National Institute of Standards and Technology, Gaithersburg, Maryland, 2006.

[10] H. Khan, M. Javed, S. Khayam and F. Mirza, Designing a cluster-based covert channel to evade disk investigation and forensics, *Computers and Security*, vol. 30(1), pp. 35–49, 2011.

[11] Microsoft, Default cluster size for NTFS, FAT and exFAT, Redmond, Washington (`support.microsoft.com/kb/140365`), 2002.

[12] Microsoft, Microsoft support policy for 4K sector hard drives in Windows, Redmond, Washington (`support.microsoft.com/kb/2510009`), 2013.

[13] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber and E. Weippl, Dark clouds on the horizon: Using cloud storage as attack vector and online slack space, *Proceedings of the Twentieth USENIX Conference on Security*, 2011.

[14] I. Thompson and M. Monroe, FragFS: An advanced data hiding technique, presented at the *BlackHat Federal Conference*, 2006.

Chapter 14

AUTOMATING VIDEO FILE CARVING AND CONTENT IDENTIFICATION

York Yannikos, Nadeem Ashraf, Martin Steinebach and Christian Winter

Abstract The massive amount of illegal content, especially images and videos, encountered in forensic investigations requires the development of tools that can automatically recover and analyze multimedia data from seized storage devices. However, most forensic analysis processes are still done manually or require continuous human interaction. The identification of illegal content is particularly time consuming because no reliable tools for automatic content classification are currently available. Additionally, multimedia file carvers are often not robust enough – recovering single frames of video files is often not possible if some of the data is corrupted or missing. This paper proposes the combination of two forensic techniques – video file carving and robust hashing – in a single procedure that can be used for the automated recovery and identification of video content, significantly speeding up forensic investigations.

Keywords: Automated forensic procedures, video file carving, robust hashing

1. Introduction

The amount of contraband image and video files has increased drastically over the past decade. As a result, forensic investigators need sophisticated tools to help recover and analyze image and video content from evidentiary sources. However, the identification of illegal content is a hard problem, and no reliable tools for automatic content classification are currently available. Thus, each multimedia file has to be inspected manually during a forensic investigation.

File carving is an advanced technique for recovering deleted data from storage devices. It works independently of the underlying file system,

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 195–212, 2013.

enabling the recovery of deleted data that has not already been overwritten.

Most audio and video files (e.g., MPEG files) are streams of data unlike WAV files, images and non-multimedia data. These streams consist of many frames and the corresponding frame headers can be used as the starting points for file carving. This makes it possible to extract individual chunks of multimedia streams when portions of the streams are already overwritten.

After multimedia data is recovered, it is necessary to conduct a time-consuming inspection and search for evidentiary data. Content classification techniques using blacklisting/whitelisting and robust hashing are often used by forensic investigators. Since blacklisting/whitelisting approaches are mostly based on cryptographic hashing, they have high false negative rates. On the other hand, robust hashing is a promising approach for detecting copies of multimedia data after content-preserving changes such as lossy compression or format conversion.

This paper proposes the combination of two forensic techniques – video file carving and robust hashing – in a single procedure that can be used for the automated recovery and identification of video content. It focuses on carving video files encoded using the MPEG video format. The procedure identifies, extracts and decodes single intracoded frames (I-frames) of MPEG video data from raw data sets in a robust manner that allows the partial recovery of frames with missing or corrupted data. Robust hashes of all the decoding results (i.e., image data) are generated and then compared with the hashes in a reference database (blacklist) to identify similarities with known illegal content. This procedure facilitates automated searches and recovery of video content, significantly speeding up forensic investigations.

2. File Carving

File carving is the process of extracting file fragments from a byte stream without file system information. This technique is mostly used to recover data from unallocated space on a hard disk. A file carver typically uses file header and footer information to identify the beginning and end of a file. Everything that lies in between a file header and footer is assumed to belong to a single file, which is then "carved" out and restored. Common file carvers that use header and footer information include PhotoRec [5], Scalpel [11] and Foremost [13]. However, if the file to be recovered is fragmented, i.e., it is not stored sequentially, but is split up into two or more parts and distributed throughout the storage area, file carving becomes difficult.

A study by Garfinkel [3] noted that, although hard disks typically exhibit low levels of fragmentation, the larger files tend to be fragmented. Since multimedia video files (e.g., AVI and MPEG files) and high resolution images (e.g., from digital cameras) are usually large, a higher percentage of them should be fragmented. In fact, Garfinkel discovered that about 17% of MPEG files and 20% of AVI files are fragmented.

Several approaches have been proposed to handle file fragmentation during file carving. One approach is bi-fragment gap carving [3], a technique that validates file contents with gaps in between them. However, this technique only works for files with no more than two fragments and with small gaps.

Pal and Memon [10] developed a graph-theoretic approach for file carving that employs hypothesis testing for fragmentation point detection. The approach is effective at recovering fragmented JPEG files. However, models and weighting techniques have to be developed for all the file types of interest.

2.1 Video File Carving

Most of the available carving strategies and implementations are not robust enough to handle multimedia files with several missing blocks. This is because missing blocks make it practically impossible to decode the data.

Since audio and video data are usually stored and transmitted as streams, the stream properties can be leveraged to support more robust data recovery and analysis. Multimedia streams comprise frames, each of which has a header followed by the actual data content. A frame header usually contains enough information to allow the calculation of the frame size, which helps determine the starting position of the next frame. This facilitates multimedia file parsing, fragmentation detection and partial recovery.

Relatively little work has been done on carving methods for video file recovery. Yoo, *et al.* [18] have developed an AVI file carving approach that works by parsing AVI file format information. They have also proposed similar approaches for MP3 and WAV files, and for NTFS compressed files. However, they assume that all the media files are stored sequentially and they only consider data cut-offs, disregarding file fragmentation with two or more fragments.

The Netherlands Forensics Institute has developed Defraser, an open source tool for video data carving [14]. Defraser can carve multimedia files such as MPEG-1/2, WMV and MP4. It can also extract and view single frames of MPEG-1 data. However, Defraser relies on a valid

MPEG-1 structure and is unable to handle small occurrences of fragmentation and missing data in specific file areas.

3. Content Identification

The identification of illegal multimedia content is an important task for forensic investigators. The typical approach is to manually inspect all the multimedia data found on a seized storage device. Automated approaches are necessary because manual inspection is very time consuming and is prone to the accidental overlooking of evidence and exonerating content.

Blacklisting and whitelisting are often used to filter known illegal and legal content. Blacklisting involves the filtering of illegal, harmful content using a reference database called the blacklist. Whitelisting involves the filtering of harmless content (e.g., operating system files) to reduce the amount of content to be investigated. To reduce the size of a reference database, an identifying hash of fixed length is generated for each file and stored instead of the file itself. Typically, cryptographic hash functions are used to generate the file hashes.

If blacklists and whitelists are available, a seized storage device can be automatically searched for known content – for each file found on the storage device, a hash is generated and compared with the hashes in the blacklist and whitelist. In this way, illegal content as well as harmless content can be identified. After this is done, it is necessary to inspect all the remaining files. This is by far the most time consuming part of a forensic investigation. When new harmful and harmless files are identified during the inspection, their hashes are stored in the blacklist and whitelist, respectively.

3.1 Robust Hashes

Unlike cryptographic hashes, "perceptual" or "robust" hashes are specifically designed for multimedia data. Robust hashes are generated from multimedia data properties based on human perception (e.g., brightness, form and contrast). Robust hashing provides an automated mechanism to decide if two multimedia files (with different cryptographic hashes) are perceived as being identical or very similar to each other. Several researchers have used robust hashing on image and video data [2, 9, 19] and on audio data [1, 6]. Lejsek, *et al.* [7] have used robust hashing to identify illegal image and video content. Their approach uses a combination of local image descriptors based on SIFT and multidimensional NV-trees to identify video data. Experiments indicate that the approach is robust to video modifications such as mirroring, scaling

and cropping. However, since the content to be identified is assumed to be available, their approach is not applicable to the preliminary investigation of storage media.

We have recently proposed an improved version [15] of the block-based video hash introduced by Yang, *et al.* [17]. This block-based video hash was chosen because it strikes the right balance between low computational complexity and good detection results.

4. Multimedia Content

Advanced methods for multimedia content recovery and identification/classification require the decoding of multimedia data. Knowledge about the specific structures of multimedia file formats is vital to these methods. Since this paper focuses on the recovery and identification of MPEG-1 video data fragments, we briefly describe the structure of the MPEG-1 video format.

An MPEG-1 video stream is essentially a sequence of still images called frames. The video stream is divided into several layers (Figure 1). The first layer is the sequence layer, which denotes a number of video sequences, each starting with a sequence header and ending with a sequence end code. The sequence header stores information necessary for decoding the actual picture data (e.g., resolution information). An arbitrary number of groups of pictures (GOPs) exist between the sequence header and sequence end code. The GOP also starts with a header followed by a number of pictures (frames). The first frame of each GOP is always an I-frame; the others are B-frames and P-frames.

In the picture layer, the header is followed by slices containing the actual picture data. The picture data itself is organized in a macroblock layer with a header, four luma blocks with brightness information and two chroma blocks with color information.

MPEG-1 files may have a video stream or video and audio streams. In the latter case, the two streams are multiplexed as a transport stream (e.g., used for network streaming) or a program stream (e.g., used for storage purposes).

5. Challenges

According to the Cyber Forensic Field Triage Process Model [12], it is important to increase the speed at which forensic analysis is performed. This requires efficient forensic tools that involve as little human interaction as possible. Automated procedures can significantly reduce the workload of forensic investigators.

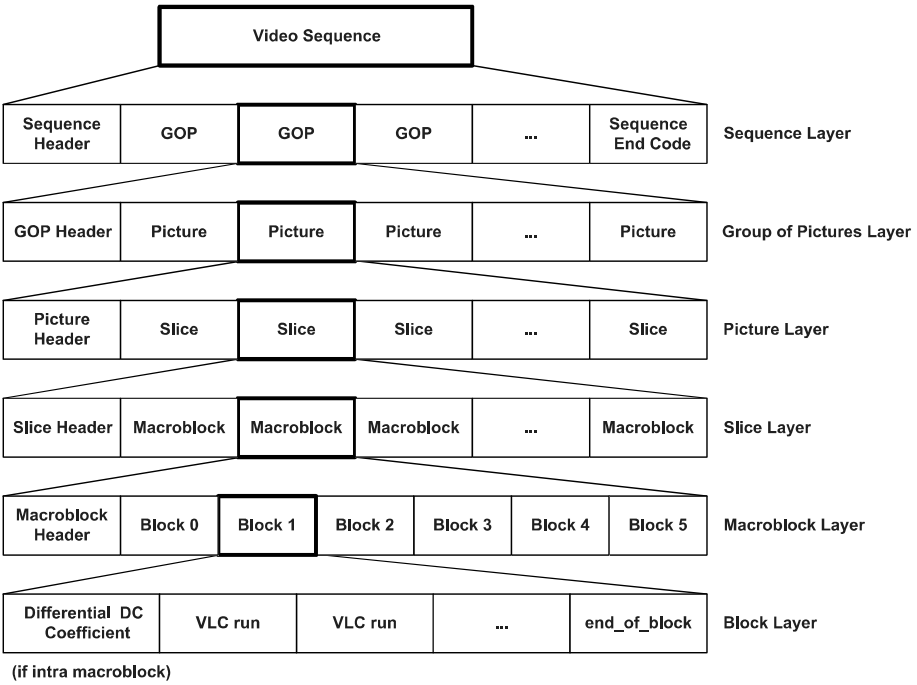


Figure 1. MPEG-1 video stream organization (from [16]).

Although several file carving approaches exist, the main challenge is dealing with fragmentation. Most file carvers are not robust enough to handle even small amounts of fragmentation, or they merely assume that all the data found between a header and footer belongs to a single file. Currently available video file carvers allow single frame carving as well as decoding and visualization, but often fail if just a few bytes of a frame are corrupted or missing.

At this time, law enforcement primarily uses cryptographic hashing to identify known multimedia content. However, this approach cannot identify content that has been (even slightly) modified, for example, through compression or cutting. For these reasons, a considerable amount of time is spent to manually inspect the various types of unidentified multimedia content that are encountered during a forensic investigation.

6. Video Frame Recovery and Identification

Our automated procedure for video file recovery and identification requires little or no interaction with a forensic investigator. Although we use the MPEG-1 as an example format for the recovery and decoding

steps, the steps can be applied to any video format with components similar to I-frames.

The automated procedure incorporates three steps that are performed in sequence:

1. **Robust Recovery:** This step involves searching a storage device or hard disk image, and reconstructing complete video files or single frames of video data found on the device. Specific properties of video file formats are engaged to support robust recovery.

2. **Robust Decoding:** This step involves the decoding of video content. Techniques such as resolution correction and resolution identification are used to enable decoding even when some of the data is corrupted or missing.

3. **Robust Identification:** This step involves the calculation of robust hashes of video content (single frames) to identify unchanged content and slightly modified content. The reference databases are also searched for known robust hashes.

6.1 Robust Recovery

The robust recovery of video content should apply file carving in a more advanced manner than traditional header/footer identification. Therefore, we designed a video frame carving approach that analyzes video frame information in order to extract and decode single I-frames. If the resolution information required for decoding is not present or cannot be found, a resolution identification/correction approach is applied to decode the available data.

The algorithm for recovering and decoding video frames has the following steps:

1. Locate all the I-frames by searching the input data for I-frame headers and verifying the headers. If no I-frame headers are found, stop.

2. Starting with the located I-frame headers, search the input data backwards for corresponding sequence headers within a specific threshold range.

3. If a sequence header is found, extract the resolution information from the sequence header; this information is needed for the subsequent decoding of I-frames.

4. Read the data following the I-frame header; this is typically optional and/or I-frame slice data. Ignore the optional data and ver-

ify the slice data. Ignore the missing and invalid slice data parts; instead, read the data that follows within a specific threshold range until additional slice data is found.

5. Read the following data and verify the structure until the next frame start code, GOP start code or sequence end code is found.

6. If the MPEG stream was identified as a system stream, demultiplex the I-frame data; otherwise, skip this step.

6.2 Robust Decoding

The recovered I-frames must be decoded to allow further processing such as automated content identification. Typically, decoding proceeds without any problem if all the information required for decoding is available. For an I-frame, this includes the resolution information, which is stored in its corresponding sequence header. If the specific sequence header is not available (e.g., due to fragmentation or overwriting), the correct resolution needed for decoding has to be found in some other manner.

Since a robust decoding should be able to handle cases where I-frame resolution information is missing, we propose the use of a resolution identification and correction procedure. This involves decoding frames with commonly used resolutions and subsequently measuring the pixel row differences of the decoded frames to automatically identify the correct decoding results. All the decoding results should be stored in a widely-supported image file format such as JPEG.

The robust decoding algorithm has the following steps:

1. Decode all the I-frames with their corresponding resolution information obtained from the sequence headers.

2. For each I-frame with missing resolution information:

   (a) Decode the I-frame using each unique resolution previously extracted from the sequence headers.

   (b) Decode the I-frame using the most common resolutions for the specific video format (e.g., source input format (SIF)).

   (c) Apply the resolution correction

3. Store each decoded result as a JPEG file.

The first two steps of the algorithm help recover and decode frames of video files that are fragmented or are partially overwritten.

Resolution Correction. During a video frame recovery process, the sequence headers containing resolution information may be missing for certain frames. This could occur because the headers were overwritten with other data or they could not be located due to fragmentation. For these frames, the following automated resolution identification/correction algorithm is applied:

1. **Identify Suitable Resolutions for Frame Decoding:** Initially identify the suitable resolutions for decoding, e.g., chosen from the most commonly used resolutions of the specific video format or from resolutions of frames found in the same raw data set.

2. **Decode Frames Using Suitable Resolutions:** Decode each frame using the previously determined suitable resolutions (individually chosen for each frame).

3. **Correct Resolution Identification for Incorrect Decoding Results:**

   (a) Divide each decoding result into blocks of 16×16 pixels.

   (b) Discard the green blocks that appear at the end of a decoding result.

   (c) Determine all possible resolutions for the remaining 16×16 pixel blocks.

   (d) Create new decoding results using these resolutions.

   (e) Measure the global pixel block row difference as follows:

$$\Delta(r) = \frac{\sum_{i=1}^{\frac{n-1}{b}} \sum_{j=1}^{m} \delta_v^2(bi, bi+1, j)}{1 + \sum_{i=1}^{\frac{n-1}{b}} \sum_{j=1}^{m} c(bi, bi+1, j)}$$

   for each decoding result r with n, m dimensions of r (n rows, m columns); b pixel block size (here, $b = 16$); $p(i, j)$ pixel value for pixel at row i, column j; vertical pixel difference $\delta_v(i, i', j) = |\, p(i, j) - p(i', j)\,|$ and

$$c(i, i', j) = \begin{cases} 1 & \text{if } \delta_v(i, i', j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

   (f) Determine the result r with the smallest $\Delta(r)$ as the correct decoding result.

6.3 Robust Identification

Robust identification is the final step of the automated procedure for processing the recovered frames. In general, the recovered video frames can be processed via automated classification or identification of their content. However, many classification approaches are somewhat limited and yield high false positive and/or false negative rates. An example is an approach that uses the amount of skin tones in an image to infer that the image shows human skin. Therefore, we propose the use of robust hashing to identify known content in an automated manner. Our robust block hashing method [15] allows for efficient content identification; it can help divide large sets of recovered frames into known and unknown content. Cryptographic hashing can also be used for automated content identification, but it lacks robustness.

Our robust block hashing method is used for the robust identification of MPEG-1 frames. Therefore, all the decoding results stored as JPEG files after the robust decoding procedure are further processed by detecting and removing horizontal or vertical black bars caused by letterbox, pillarbox or windowbox effects. Then, the corresponding robust block hashes are calculated, which are subsequently used to query a reference database (e.g., with blacklisted content). If the same or a similar robust block hash is found in the reference database, the corresponding I-frame is classified as showing illegal content.

7. Experimental Evaluation

This section describes the experimental results obtained for the robust recovery, robust decoding and robust identification steps of our procedure.

7.1 Recovery (MPEG-1 Frame Carving)

For our experiments, we prepared ten data sets, each consisting of a raw byte stream with random data. Five MPEG-1 files with only video data were chosen (i.e., "elementary stream" MPEG-1 files), and five files with multiplexed video and audio data were chosen (i.e., "system stream" MPEG-1 files). The MPEG-1 files had a frame rate of 25 or 29.9 fps, bit rates ranging from 0.58 to 4.09 Mbps, file sizes between 3 MiB and 46 MiB; and contained between 29 and 302 I-frames.

For each data set, we took one of the MPEG-1 files, split a randomly (non-zero) sized portion of the file into chunks of predefined size s, and put these chunks into the data set in a non-consecutive order. The first group of five data sets G_{elem} comprised the elementary stream MPEG-1

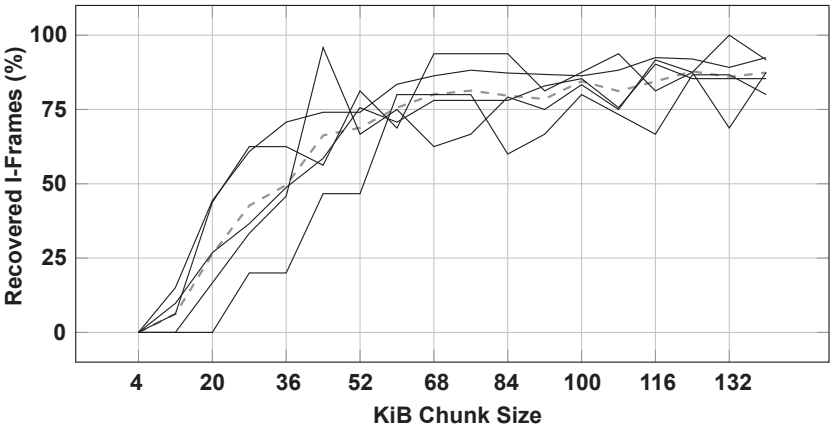


Figure 2. Percentage of completely recovered I-frames per chunk size for G_{elem}.

files and the second group of five data sets G_{sys} comprised the system stream MPEG-1 files. No MPEG-1 file was used twice. The resulting data set essentially corresponds to a hard drive image filled with random data and a single fragmented MPEG-1 file with s-sized fragments. The data set preparation was performed using standard Unix tools.

After the data sets were prepared, our MPEG-1 video frame carving prototype was used to search each data set for I-frames to be reconstructed. The number of I-frames that were completely or partly reconstructed was recorded. After each test run, the chunk size s was increased, and ten new data sets were prepared by filling them with random data and using the same set of MPEG-1 files. Then, the next test run was started. Eighteen different chunk sizes ranging from 4 to 140 KiB were used; thus, each data set was tested eighteen times.

The experimental results in Figures 2 and 3 show that the number of completely recovered I-frames grows with increasing chunk size, which is to be expected. Figure 2 shows that, for a mean chunk size of at least 36 KiB, nearly 50% (49.57%) of the I-frames could be completely recovered from G_{elem}. On the other hand, Figure 3 shows that a mean chunk size of 44 KiB was necessary to completely recover at least 50% (50.25%) of the I-frames from G_{sys}.

Figure 2 shows the percentage of completely recovered I-frames per chunk size for the five data sets in G_{elem} containing chunks of elementary stream MPEG-1 video files (mean values shown as a dashed line). Note that the highest mean rate of 87.8% completely recovered I-frames from G_{elem} was obtained for a chunk size of 124 KiB, which is roughly equal to the size of a GOP with two additional average-sized I-frames.

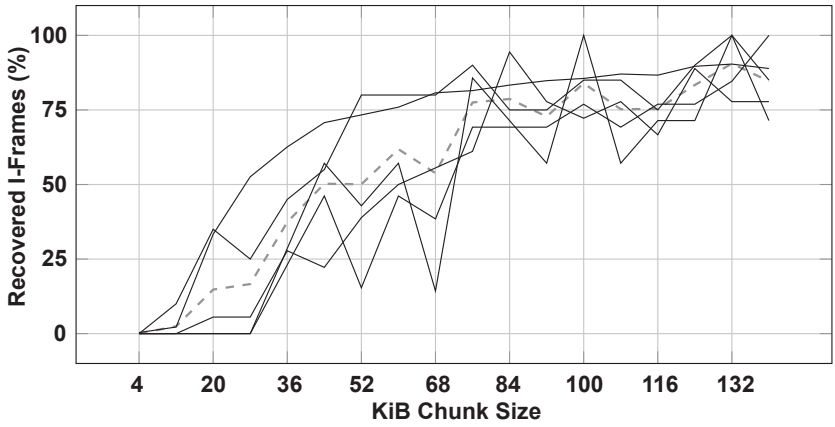


Figure 3. Percentage of completely recovered I-frames per chunk size for G_{sys}.

Figure 3 shows the percentage of completely recovered I-frames per chunk size for the five data sets in G_{sys} containing chunks of system stream MPEG-1 video files (mean values shown as a dashed line). Note that the highest mean rate of completely recovered I-frames was 90.55% for a chunk size of 132 KiB.

The different results obtained for G_{elem} and G_{sys} are due to the different file structures. The MPEG-1 files constituting the system stream contain more header information as well as audio data. Therefore, larger chunk sizes are needed to obtain better carving results.

The results show that I-frames from fragmented MPEG-1 files even with relatively small chunks (fragments) can be recovered completely or partially. Starting with a chunk size of 28 KiB, 20% or more of the I-frames were recovered completely or partially. Note that both the completely and partially recovered I-frames (which cause incorrect decoding results) are useful to evaluate the robust hashing method.

We also compared the performance of our MPEG-1 video frame carving implementation with Defraser [14], which works in a similar manner. For the comparison, we prepared six MPEG-1 video files (25 or 29.9 fps, 1.13 to 3.05 Mbps bit rate, 6 MiB to 46 MiB file size) and cut them into chunks of 4, 8, 16, 32, 64, 128 and 256 KiB. The chunks were then placed on a 1 GiB hard disk image file that was previously filled with random data.

Our prototype implementation and Defraser were able to recover 92% of the 361 I-frames. Of the recovered frames, both Defraser and our implementation were able to recover 82% I-frames completely and 18% partially. However, because Defraser relies on finding intact sequence headers, it was unable to recover I-frames with corrupted or missing

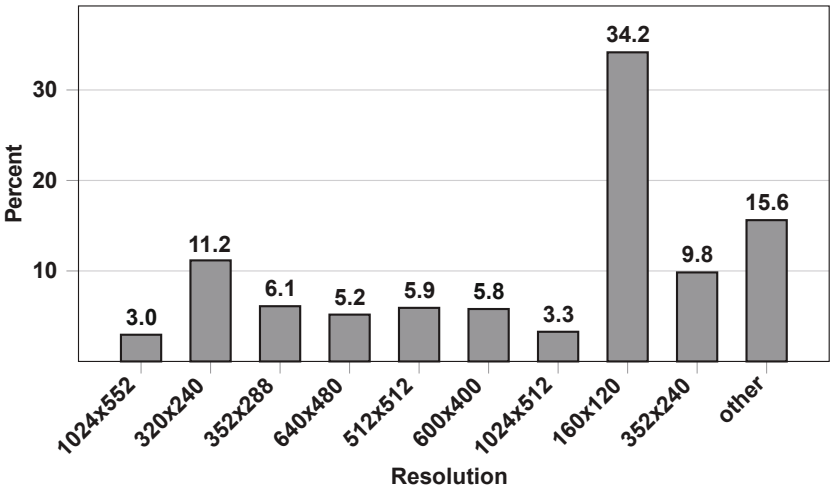


Figure 4. Resolution distribution of 5,489 randomly-downloaded MPEG-1 video files.

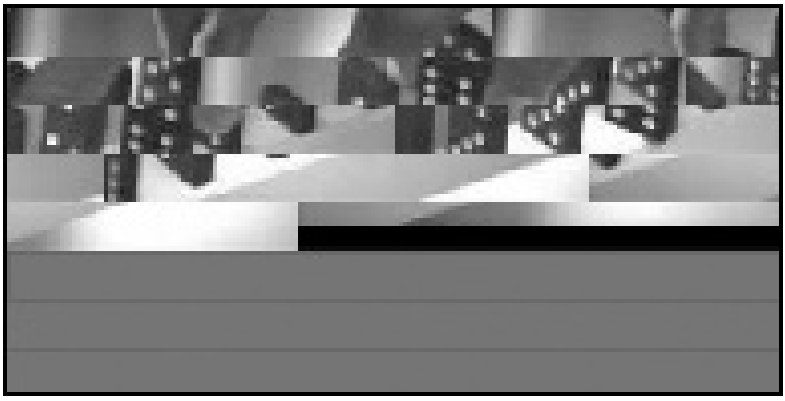
sequence headers. For such cases, we used the frame resolution correction procedure, which is described below.

7.2 Decoding (Resolution Correction)

A post-processing step is required to find the correct resolution when the resolution information of individual frames is missing. The post-processing was done as follows:

- **Identification of Resolutions for Frame Decoding:** First, we identified the most commonly used resolutions of the MPEG-1 video format by collecting 5,489 different MPEG-1 videos from random sources using Google. Next, we extracted the resolutions that were used. We observed that nearly 85% of the video files used one of nine resolutions, including the SIF resolutions. Note that, according to the MPEG specification, MPEG-1 video data should be encoded using one of three SIF resolutions: 352×240, 352×288 or 320×240 pixels. Figure 4 shows the resolution distribution of the MPEG-1 files. We believe that, together with the most commonly used MPEG-1 video resolutions, all the resolutions of frames found in the same raw data set would be suitable for individual frame decoding and subsequent resolution identification.

- **Frame Decoding and Resolution Correction:** Using the resolutions that were determined to be suitable, we decoded each frame several times, each time with a different resolution. The results were stored as JPEG files. For each frame, we chose the decod-



(a) Original frame with incorrect resolution.

(b) Same frame with corrected resolution.

Figure 5. Frame decoding results before and after resolution correction.

ing result with the least amount of green blocks and applied our resolution correction approach. To evaluate the resolution correction approach, we used a test set of ten sample frames, where each frame was stored using 30 different resolutions. Upon applying the resolution correction, we were able to automatically produce decoding results with correct resolutions for about 80% of the JPEG files. Figure 5 shows an example of a frame decoding result with incorrect resolution and one with corrected resolution.

7.3 Identification (Robust Hashing)

With regard to content identification, we used our prototype implementation as reported in [15] to evaluate robust block hashing with a

test set containing simulated illegal images. Specifically, the test set contained 4,400 images of a cheerleader team, the notable characteristics being a large quantity of skin colors, similar poses in all the images, and the presence of one or more persons in all the images. The images were randomly divided into two equal-sized groups. One group was used as a test set to be recognized and was added to a robust hash database that already contained 88,000 robust hashes of other images. The second group of images was used as a test set to evaluate the false positive rate.

All 4,400 images were scaled down by a factor to ensure that the larger edge was only 300 pixels long; this corresponded to an average size reduction of 25% compared with the original images. Next, the images were horizontally mirrored and stored with a JPEG quality factor of 20, producing small images of low quality. This procedure simulates the strong changes that can occur during image conversion.

The two groups of images could be distinguished rather well using the Hamming distance between robust hashes as a feature to distinguish between known and unknown images. With a Hamming distance of 32 as the threshold, a false positive rate of 1.1% and a false negative rate of 0.4% were obtained. When using a Hamming distance of eight as the threshold, the false negative rate increased to 10% while the false positive rate dropped to 0%. Using both the Hamming distance and a weighted distance as proposed in [15] yielded even better results: for a Hamming distance of eight and a weighted distance of sixteen, the false positive rate dropped to 0% and the false negative rate fell to just 0.2%.

We also used our implementation to measure the time needed to create a robust hash of an image and to look up the hash in a database. We used the same image set of 4,400 images and the same robust hash database (88,000 plus 2,200 hashes). Our experiments revealed that each image was processed in roughly 10 ms, including robust hash generation and database lookup. We also observed that the speed could be increased up to factor of four when hard drive caching was used. These results are comparable with those obtained when computing cryptographic hashes (e.g., SHA-1) of the image files.

8. Conclusions

Combining robust video file recovery, robust frame decoding and robust video content identification into a single automated procedure can speed up forensic investigations. The prototype implementation described in this paper can recover and decode single MPEG-1 I-frames in the presence of small amounts of fragmentation. Additionally, ro-

bust block hashing can help recognize known image material, such as recovered and decoded I-frames, with low error rates and computational overhead.

The prototype implementation provides good results with regard to recovery, decoding and identification. Most video file carving approaches do not work well in the presence of fragmentation. Additionally, unlike our prototype, most carving tools do not have integrated I-frame decoders. The Defraser tool [14] can decode and visualize I-frames, but fails if just a few bytes of data (e.g., sequence header) needed for decoding are not available. Further research is necessary on reliable fragmentation point detection for multimedia file formats. Pal, *et al.* [10] have proposed a promising approach for text and image recovery, especially for JPEG files. However, fragmentation point detection approaches have yet to be developed for a wide range of video and audio file formats.

Robust hashing clearly improves video content identification mechanisms such as cryptographic hashing, which are prone to fail if the data to be hashed has been manipulated even slightly. Also, block-wise cryptographic hashing as proposed in [4] can help identify known video content, especially single frame fragments that cannot be decoded anymore. However, since the corresponding reference database can be very large, it is necessary to explore approaches for trading-off fragmentation robustness, database size and lookup speed.

Acknowledgement

This research was supported by the Center for Advanced Security Research Darmstadt (CASED), Darmstadt, Germany.

References

[1] E. Allamanche, J. Herre, O. Hellmuth, B. Froba, T. Kastner and M. Cremer, Content-based identification of audio material using MPEG-7 low level description, *Proceedings of the Second International Symposium on Music Information Retrieval*, 2001.

[2] J. Fridrich and M. Goljan, Robust hash functions for digital watermarking, *Proceedings of the International Conference on Information Technology: Coding and Computing*, pp. 178–183, 2000.

[3] S. Garfinkel, Carving contiguous and fragmented files with fast object validation, *Digital Investigation*, vol. 4(S), pp. S2–S12, 2007.

[4] S. Garfinkel, A. Nelson, D. White and V. Roussev, Using purpose-built functions and block hashes to enable small block and sub-file forensics, *Digital Investigation*, vol. 7(S), pp. S13–S23, 2010.

[5] C. Grenier, PhotoRec (`www.cgsecurity.org/wiki/PhotoRec`), 2007.

[6] J. Haitsma, T. Kalker and J. Oostveen, Robust audio hashing for content identification, *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, pp. 117–124, 2001.

[7] H. Lejsek, A. Johannsson, F. Asmundsson, B. Jonsson, K. Dadason and L. Amsaleg, Videntifier forensics: A new law enforcement service for the automatic identification of illegal video material, *Proceedings of the First ACM Workshop on Multimedia in Forensics*, pp. 19–24, 2009.

[8] N. Memon and A. Pal, Automated reassembly of file fragmented images using greedy algorithms, *IEEE Transactions on Image Processing*, vol. 15(2), pp. 385–393, 2006.

[9] J. Oostveen, T. Kalker and J. Haitsma, Visual hashing of video: Application and techniques, *Proceedings of the Thirteenth IS&T/SPIE International Symposium on Electronic Imaging, Security and Watermarking of Multimedia Contents*, vol. 4314, 2001.

[10] A. Pal, H. Sencar and N. Memon, Detecting file fragmentation points using sequential hypothesis testing, *Digital Investigation*, vol. 5(S), pp. S2–S13, 2008.

[11] G. Richard III and V. Roussev, Scalpel: A frugal, high performance file carver, *Proceedings of the Fifth Annual Digital Forensics Research Workshop*, 2005.

[12] M. Rogers, J. Goldman, R. Mislan, T. Wedge and S. Debrota, Computer Forensics Field Triage Process Model, *Proceedings of the Conference on Digital Forensics, Security and Law*, pp. 27–40, 2006.

[13] SourceForge.net, Foremost(`foremost.sourceforge.net`).

[14] SourceForge.net, NFI Defraser (`sourceforge.net/projects/defraser`).

[15] M. Steinebach, H. Liu and Y. Yannikos, Forbild: Efficient robust image hashing, *Proceedings of the SPIE Conference on Media Watermarking, Security and Forensics*, vol. 8303, 2012.

[16] S. Vimal, Introduction to MPEG Video Coding, Lectures on Multimedia Computing, Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, India, 2007.

[17] B. Yang, F. Gu and X. Niu, Block mean value based image perceptual hashing, *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 167–172, 2006.

[18] B. Yoo, J. Park, S. Lim, J. Bang and S. Lee, A study of multimedia file carving methods, *Multimedia Tools and Applications*, vol. 61(1), pp. 243–251, 2012.

[19] X. Zhou, M. Schmucker and C. Brown, Video perceptual hashing using interframe similarity, *Proceedings of the 2006 Sicherheit Conference*, pp. 107–110, 2006.

Chapter 15

DATA RECOVERY FROM PROPRIETARY-FORMATTED CCTV HARD DISKS

Aswami Ariffin, Jill Slay and Kim-Kwang Choo

Abstract Digital video recorders (DVRs) for closed-circuit television (CCTV) commonly have an in-built capability to export stored video files to optical storage media. In the event that a DVR is damaged, its contents cannot be easily exported. This renders the forensically-sound recovery of proprietary-formatted video files with their timestamps from a DVR hard disk an expensive and challenging exercise. This paper presents and validates a technique that enables digital forensic practitioners to carve video files with timestamps without referring to the DVR hard disk filesystem.

Keywords: Digital CCTV forensics, video stream, data carving

1. Introduction

Video surveillance and closed-circuit television (CCTV) systems serve as deterrents to crime, and can be used to gather evidence, monitor the behavior of known offenders and reduce the fear of crime [1]. CCTV systems can be broadly categorized into analog, digital and Internet Protocol (IP) based systems. Analog systems have limited abilities to store, replicate and process large amounts of video data, and the quality of images and video files is generally quite low. Digital CCTV systems use digital cameras and hard disk storage media. IP based CCTV systems stream digital camera video using network protocols.

The primary goal of digital CCTV forensics is to ensure the reliability and admissibility of video evidence [4]. To reduce the risk of digital evidence being called into question in judicial proceedings, it is important to have a rigorous methodology and procedures for conducting digital forensic examinations. For example, in a case involving evidence obtained from a digital CCTV system, one of the first activities is to

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 213–223, 2013.

recover video evidence from the storage media in a forensically-sound manner.

In digital CCTV forensics, it is extremely challenging to recover evidence in a forensically-sound manner without data recovery expertise in a wide range of storage media with different filesystems and video formats. The challenge is compounded if the storage media of a digital video recorder (DVR) is damaged or corrupted. The variety of digital CCTV systems further complicates the digital forensic process, as many of the systems use proprietary technologies. Therefore, digital forensic practitioners need to have an intimate understanding of digital CCTV systems.

This paper describes a forensically-sound technique for carving video files with timestamps without referring to the DVR hard disk filesystem. The technique is validated using a customized DVR with a proprietary file format.

2. Digital CCTV Forensics

The digital forensic framework of McKemmish [2] has four steps: (i) identification; (ii) preservation; (iii) analysis; and (iv) presentation. The proposed digital CCTV forensic technique shown in Figure 1, which extends McKemmish's framework, incorporates four broadly similar steps: (i) preparation; (ii) cloning; (iii) analysis; and (iv) reporting. The following sections discuss the details underlying digital CCTV forensics with respect to the steps in McKemmish's framework.

2.1 Step 1: Identification

The identification step involves a study of the device to understand what evidence is likely to be present, its location and technical specifications. For example, in cases involving a personal computer, the size of the potential evidentiary data can be huge and could include audio and video files, images, Internet browsing histories, emails and GPS data.

The first action that a digital forensic practitioner should take is to conduct a visual inspection of the exhibit to check for damage and, if possible, record the system time offset (difference between the actual time and the device time).

In cases involving a digital CCTV system, a digital forensic practitioner will need to understand its main electronic components. In a digital CCTV system, the camera captures the scene using a charge-coupled device and converts it to electrical signals. The signals are converted to digital data by the DVR using an analog-to-digital converter. The video data is then formatted in a file container with a specific video codec and

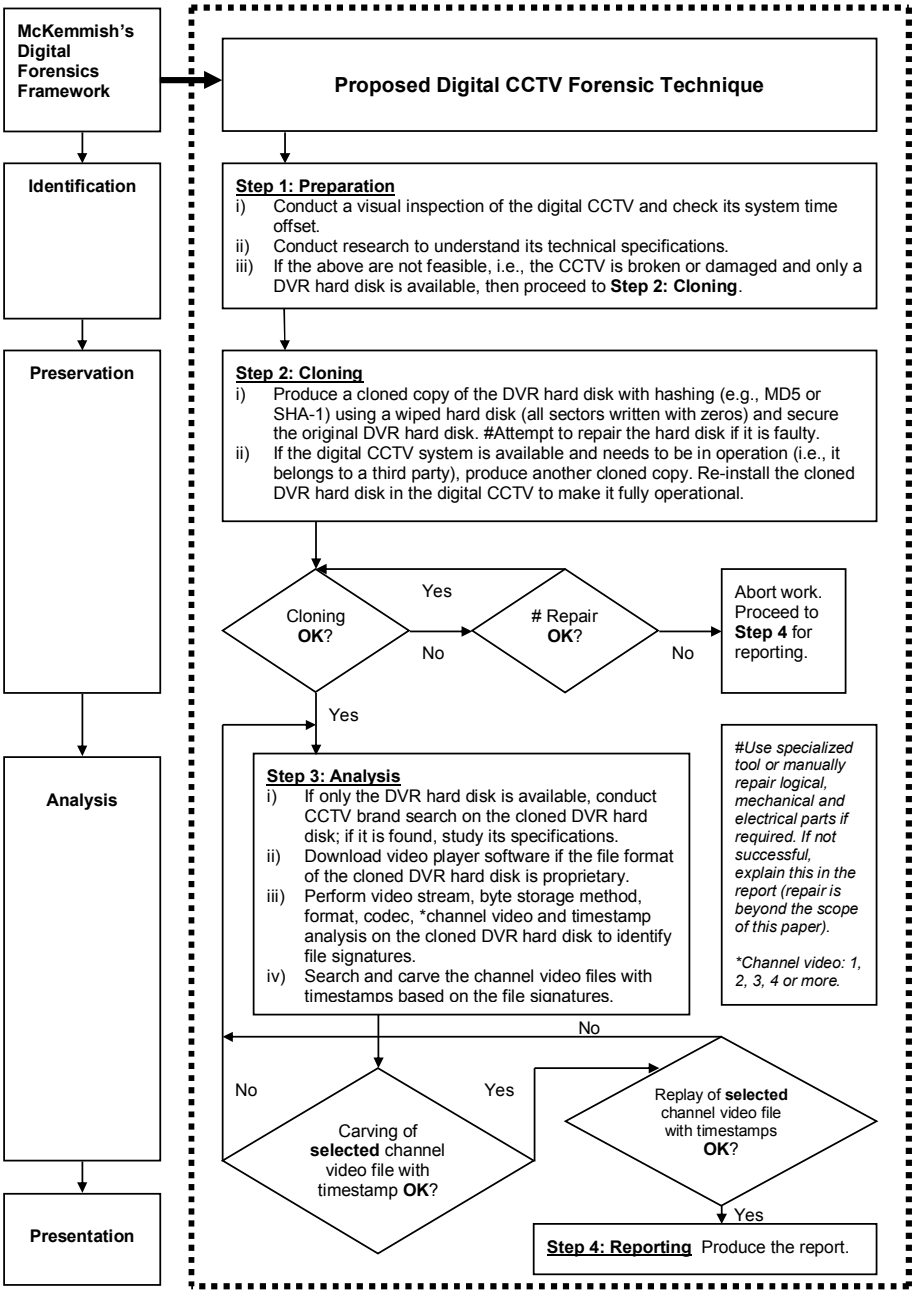


Figure 1. Proposed digital CCTV forensic technique.

other digital data such as timestamps and audio. Finally, the video file is stored on a hard disk.

During the identification step, it may be determined that additional expertise or resources are required to ensure that the forensic team has the requisite knowledge and tools to recover the data. If the digital CCTV system is damaged and it is not possible to conduct a visual inspection and record the time offset, the digital forensic practitioner may proceed to the next step.

2.2 Step 2: Preservation

The preservation step seeks to ensure that the evidentiary data remains unchanged. Given the likelihood of judicial scrutiny, it is imperative that the data be recovered using a forensically-sound method (e.g., it should not write to the original data source). In the event that the storage media is damaged or corrupted, it is necessary to repair the media before the acquisition can commence.

Next, a bit-for-bit copy of the DVR hard disk is created. This copy (image) is hashed and the hash value is retained to prove that it is a duplicate copy of the original media.

A cloned copy of the DVR hard disk may have to be produced if the system belongs to a third party or if the system must remain operational. A digital CCTV system may not belong to the suspect and seizing the system is usually a last resort. Cloning rather than imaging the DVR hard disk is the preferred approach because an imaged hard disk cannot be replayed or made to record video like a cloned copy.

A live cloned copy must also be created if the DVR system has RAID storage. Duplicating the media simplifies the analysis task, eliminating the need for specialized tools and knowledge about the RAID configuration.

2.3 Step 3: Analysis

The third step is to analyze the cloned hard disk, examining the video stream, byte storage method, format, codec, channel and timestamp to identify the file signatures for searching and carving the video files with timestamps. The timestamps of the video files can be obtained from the filesystem using standard recovery procedures. The proposed technique accounts for the possibility that the filesystem is unrecognized by the standard recovery tools and is able to recover the timestamps.

A key challenge to recovering video files with timestamps from a DVR hard disk with a proprietary file format is to analyze the video stream for its internal attributes without the assistance of technical documentation.

However, video file formats have a header and occasionally a footer that provide file metadata, including the format of the video stream. Using this information, a digital forensic practitioner can select the appropriate player and video codec to replay the video files with timestamps. Therefore, we suggest building a database of known file signatures that can be used by a digital CCTV forensic software suite.

For a DVR hard disk that uses a proprietary file format, there are some technical issues that need special attention. First, it is necessary to determine if the byte storage method is little endian or big endian. After the byte storage method has been determined, the file signatures can be derived and this information can be used to correlate each file signature to the channel video that captured the scenes with timestamps.

If the video codec is also proprietary, a proprietary software player must be downloaded from the manufacturer's website. If a player is not available, further research must be conducted on the codec because the carved video files with timestamps cannot be replayed using a standard video player and codec.

2.4 Step 4: Reporting

The fourth step is to ensure that the findings are presented in a manner that is understandable to investigators, prosecutors, members of the judiciary, and other decision makers. When a number of parties are involved in evidence recovery, it is critical that the chain of custody be logged, along with details about the individuals who repaired the device and/or extracted data, and any changes that were made as part of these procedures.

3. Application of the Forensic Process

Table 1 lists the tools and software used in the forensic analysis. Although the CCTV brand was known to be RapidOS, the following use case discusses how brand information can be determined.

3.1 Identification

The first step is to check the DVR time setting and determine the offset from the actual time. The time offset is beneficial for verification against the actual time of the incident. In this particular example, we proceed to the second step (Preservation) given there are no constraints that the device must remain in operation.

Table 1. Tools and software.

Number	Item
1	EnCase 6.7 and FTK 3.0, widely-used commercial digital forensic tools.
2	WinHex 14.5, hexadecimal editor tool commonly used for data recovery and digital forensics.
3	RapidOS T3000-4 Viewer software, proprietary video, audio and timestamp player.
4	MacBook Pro laptop with Windows XP virtual system.
5	500 GB DVR hard disk used in a RapidOS digital CCTV system with a unit of the same size hard disk for cloning purposes (sectors were wiped with zeros).
6	Tableau Forensic Bridge T35es-R2, used to prevent writing on the source hard disk during cloning.
7	DCode v.4.02a, used for Unix 32-bit hexadecimal time conversion.

3.2 Preservation

For preservation purposes, the DVR hard disk was cloned and its hash value was computed using WinHex 14.5. The size of the cloned hard disk was 500 GB and the time taken to create the image was about 27 hours. In a real-world situation, two clones are required – one for forensic analysis and the other for re-installation into the digital CCTV system for continuous operation. The original hard disk is then retained in an evidence preservation facility for future reference.

The MD5 hashing process took approximately nine hours. The image was then verified as accurate.

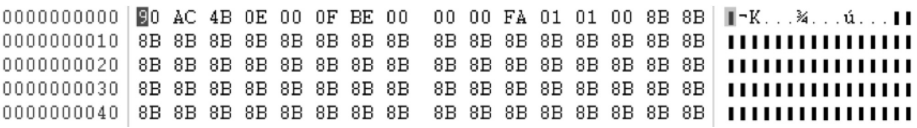


Figure 2. Proprietary file format.

3.3 Analysis

EnCase 6.7 and FTK 3.0 were used to check if the filesystem was readable. Both tools were unable to recognize the filesystem and the video files. Further analysis used WinHex 14.5 to perform byte-level analysis. As shown at offset `0` in Figure 2, the file format was unknown and, therefore, proprietary. Unlike Poole, *et al.* [3], we did not conduct further forensic analysis of the filesystem of the cloned DVR hard disk

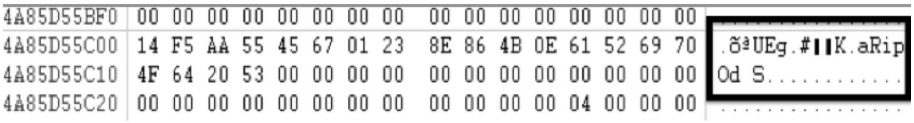


Figure 3. Confirmation of CCTV brand and byte storage method.

because it would have been time consuming. In any case, this was impractical because we did not have access to a complete digital CCTV system.

Next, each sector (512 bytes) of the disk was analyzed to determine the CCTV brand. We found the text string "aRipOd S" at offset `4A85D55C00`, which identified the digital CCTV system brand as RapidOS. The text arrangement was not directly matched because of its byte storage method. The RapidOS system stores data in a reverse 16-bit byte little endian format (Figure 3). This finding was useful for further forensic analysis of the unknown content format.

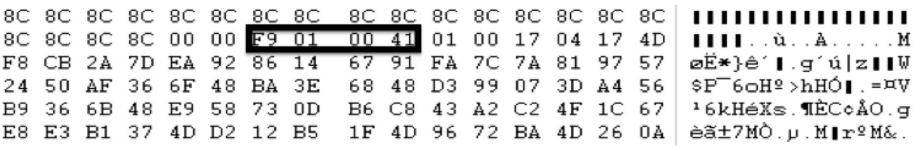


Figure 4. Example of channel video file signature.

Analysis then focused on recognizable file signatures. For this type of CCTV, it involved required searching for repetitive hexadecimal patterns in the channel video (from one to the number of channels) interleaved with timestamp tracks:

- \0xF9\0x01\0x00\0x40\
- \0xF9\0x01\0x00\0x41\ (Example shown in Figure 4.)
- \0xF9\0x01\0x00\0x42\
- \0xF9\0x01\0x00\0x43\
- \0xFA\0x01\0x01\0x00\
- \0xF0\0x7E\0x4B\0x0B\ (According to our findings, the first two bytes change frequently in successive tracks.)

Based on the repetitive hexadecimal pattern, the T3000-4 Viewer software and manual was downloaded to gain additional technical information about the digital CCTV system. Because of the proprietary

nature of the video format, the technical specification did not provide enough details to develop tools for further analysis. Additionally, only the T3000-4 Viewer software played the VVF file format found on the drive. Overall, 50 GB of data for all four channels of CCTV videos with timestamps were extracted from the drive image. Viewing the video confirmed that the cloned DVR hard disk was a four-channel digital CCTV system and allowed for further analysis of the repetitive interleaving hexadecimal signatures. After performing the analysis, the video and timestamp file signatures were interpreted as follows:

- \0x01\0xF9\0x40\0x00\ → Channel 1 video header file signature
- \0x01\0xF9\0x41\0x00\ → Channel 2 video header file signature
- \0x01\0xF9\0x42\0x00\ → Channel 3 video header file signature
- \0x01\0xF9\0x43\0x00\ → Channel 4 video header file signature
- \0x01\0xFA\0x00\0x01\ → Footer file signature
- \0x7E\0xF0\0x0B\0x4B\ → Unix 32-bit hexadecimal timestamp (little endian)

Further forensic analysis was performed on the cloned hard disk content to demonstrate that the proposed technique was capable of searching and carving on selected channel video and timestamps. This can considerably shorten the time required for a forensic examination of a digital CCTV system. The selected channel video and timestamp details were:

- Channel 1 video header file signature → \0x01\0xF9\0x40\0x00\
- Footer file signature → \0x01\0xFA\0x00\0x01\
- Timestamp: 24th November 2009 at 14:41:01 → \0x7D\0xF0\0x0B\0x4B\

We converted the timestamp to Unix 32-bit hexadecimal using DCode v.4.02a in order to search the cloned DVR hard disk.

The Channel 1 video header, footer (file signatures) and Unix 32-bit hexadecimal timestamp were converted to the RapidOS 16-bit byte little endian format:

- \0xF9\0x01\0x00\0x40\ → Channel 1 video header file signature
- \0xFA\0x01\0x01\0x00\ → Footer file signature

Figure 5. Sample channel video file image with timestamp.

- \0xF0\0x7D\0x4B\0x0B\ → Timestamp of November 24, 2009 at 14:41:01

WinHex 14.5 was then used to search and carve out the digital video evidence according to the channel (video header and footer) and timestamp signatures. The channel video file with timestamp was 16-bit byte swapped and converted into VVF file format for replay using the T3000-4 Viewer (Figure 5). Note that some of the video has been intentionally darkened to mask the identities of the individuals.

3.4 Reporting

For the findings of the forensic analysis to be meaningful, they must be reported in a manner that complies with the local court requirements. The report may be verbal or written and should include information on all procedures, tools and software used along with their limitations to prevent false conclusions from being reached.

4. Forensic Analysis of a Customized DVR

A forensic analysis of an AVTECH digital CCTV KPD674 4-Channel DVR customized with a database was conducted as an additional test to provide support for the proposed technique. The AVTECH DVR recordings were made for five days over three months. The database consisted of several records, primarily an image identity number, offset and Unix timestamp records. SQLite Database Browser 2.0b1 and MPlayer 06.9+SVN-r3607 were used as additional software during our analysis.

The DVR file system was intentionally corrupted to demonstrate that video files with timestamps were recoverable. Recovery involved searching for the database file signature: \0x53\0x51\0x4C\0x69\0x74\0x65\0x20\0x66\0x6F\0x72\0x6D\0x61\0x74\0x20\0x33\0x00\ and subsequently extracting the SQLite database file. Concurrently, adjacent raw hexadecimal video streams were analyzed to identify the video codec and then extracted. The video codec was identified as H.264 from the hexadecimal file signature of \0x32\0x36\0x34\. Other video specifica-

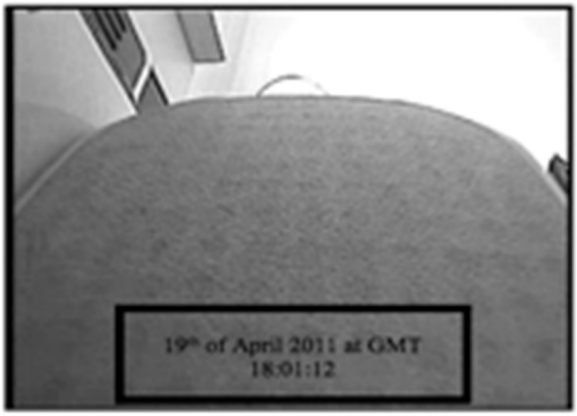


Figure 6. Customized AVTECH DVR video snapshot using MPlayer.

tions such as resolution, aspect ratio and frames per second can also be obtained from the video header data.

MPlayer 06.9+SVN-r3607 was able to replay the H.264 video files. The timestamp was obtained from the database using SQLite Database Browser 2.0b1. Figure 6 shows a snapshot taken from the AVTECH DVR video stream. The timestamp from the database was April 19, 2011 at GMT 18:01:12.

5. Conclusions

The digital CCTV forensic technique presented in this paper enables video files with timestamps to be carved without referring to the filesystem. The technique is designed to be independent of the brand, model, media and filesystem. Moreover, it does not require interaction with the vendor or manufacturer.

Our future work will focus on developing a digital CCTV forensic software suite. A video-carving tool will be included in the software suite. The tool will be intuitive and easy to use. It will help create forensic copies and reference lists for various video headers and footers (file signatures), and will search for and carve video files with timestamps with minimal human intervention.

References

[1] K. Choo, Harnessing information and communications technologies in community policing, in *Community Policing in Australia*, J. Putt (Ed.), Australian Institute of Criminology, Canberra, Australia, pp. 67–75, 2010.

[2] R. McKemmish, What is forensic computing? *Trends and Issues in Crime and Criminal Justice*, no. 118, 1999.

[3] N. Poole, Q. Zhou and P. Abatis, Analysis of CCTV digital video recorder hard disk storage system, *Digital Investigation*, vol. 5(3-4), pp. 85–92, 2009.

[4] G. Porter, A new theoretical framework regarding the application and reliability of photographic evidence, *International Journal of Evidence and Proof*, vol. 15(1), pp. 26–61, 2011.

V

NETWORK FORENSICS

Chapter 16

CREATING INTEGRATED EVIDENCE GRAPHS FOR NETWORK FORENSICS

Changwei Liu, Anoop Singhal and Duminda Wijesekera

Abstract Probabilistic evidence graphs can be used to model network intrusion evidence and the underlying dependencies to support network forensic analysis. The graphs provide a means for linking the probabilities associated with different attack paths with the available evidence. However, current work focused on evidence graphs assumes that all the available evidence can be expressed using a single, small evidence graph. This paper presents an algorithm for merging evidence graphs with or without a corresponding attack graph. The application of the algorithm to a file server and database server attack scenario yields an integrated evidence graph that shows the global scope of the attack. The global graph provides a broader context and better understandability than multiple local evidence graphs.

Keywords: Network forensics, probabilistic evidence graphs, attack graphs

1. Introduction

Evidence graphs built from network attacks can be used by network forensic practitioners to model evidence of network intrusions [13]. The nodes in an evidence graph represent computers that are of interest in a network forensic investigation and the edges represent the dependencies between evidentiary items.

Attack graphs have been used to analyze security vulnerabilities and their dependencies in enterprise networks. By incorporating quantitative metrics, probabilistic attack graphs can help estimate the probabilities of successful network attacks [11]. While a good probabilistic attack graph provides attack success probabilities to assist forensic investigations and network configuration adjustments, an attack graph with inaccurate attack paths or attack success probabilities can mislead in-

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 227–241, 2013.

vestigators and network administrators. To address this limitation, Liu, *et al.* [6] have proposed the adjustment of an inaccurate probabilistic attack graph by mapping a corresponding probabilistic evidence graph to the attack graph. Their algorithm can map an evidence graph to a small-scale attack graph, but it does not provide a solution for mapping multiple evidence graphs to a single large-scale attack graph. The reason for having multiple attack graphs is clear – evidence is collected from multiple systems and digital forensic practitioners must combine the individual graphs to obtain a complete view of the evidence trail left by a distributed attack. To the best of our knowledge, the problem of integrating evidence graphs has not been addressed by other researchers.

2. Background

Sheyner, *et al.* [10] have defined state-based attack graphs whose nodes correspond to global states of a system and edges correspond to state transitions. However, a state-based attack graph can yield an exponential number of states because it encodes nodes as a collection of Boolean variables to cover the entire set of possible network states. A more compact graph representation expresses exploits or conditions as nodes and attack dependencies as edges [1, 4, 9]. For example, Wang and Daniels [13] have defined the notion of an evidence graph whose nodes represent host computers involved in an attack and edges represent forensic evidence that correlate the hosts.

The NIST National Vulnerability Database (NVD) [8] standardizes vulnerability metrics that assign success probabilities to exposed individual vulnerabilities. The NVD provides the probabilities of attacks used in attack graphs [9, 12]. Evidence graphs also use quantitative metrics to estimate the admissibility of evidence and the certainty that a particular host is involved in an attack [13].

Definition 1. Evidence Graph [6, 13]: An evidence graph is a sextuple $G = (N, E, N - Attr, E - Attr, L, T)$ where N is a set of nodes representing host computers, $E \subseteq (N_i \times N_j)$ is a set of directed edges each consisting of a particular data item that indicates activities between the source and target machines, $N - Attr$ is a set of node attributes that include the host ID, attack states, timestamp and host importance, and $E - Attr$ is a set of edge attributes each consisting of an event description, evidence impact weight, attack relevance and host importance. Functions $L : N \rightarrow 2^{N-Attr}$ and $T : E \rightarrow 2^{E-Attr}$ assign attribute-value pairs to nodes and edges, respectively.

The $N - Attr$ attack states comprise one or more of attack source, target, stepping-stone and affiliated host computers. Affiliated hosts are

computers that have suspicious interactions with an attacker, a victim or stepping-stone host [6].

Definition 2. Probabilistic Evidence Graph [6]: In a probabilistic evidence graph $G = (N, E, N - Attr, E - Attr, L, T, p)$, the probability assignment functions $p \in [0, 1]$ for an evidence edge $e \in E$ and a victim host $n \in N$ are defined as:

- $p(e) = c \times w(e) \times r(e) \times h(e)$ where w, r and h are the weight, relevance and importance of the evidence edge e, respectively [13]. The coefficient c indicates the category of evidence, which includes primary evidence, secondary evidence and hypothesis testing from expert knowledge that are assigned values of 1, 0.8 and 0.5, respectively.

- $p(n) = p[(\cup e_{out}) \cup (\cup e_{in})]$ where $\cup e_{out}$ contains all the edges whose source computer is a host n with a particular attack-related state, and $\cup e_{in}$ contains all the edges whose target computer is n with the same state.

The weight of an evidence edge with a value in $[0, 1]$ represents the impact of the evidence on the attack. The relevance is the measure of the evidence impact on the attack success at this specific step, which can take one of three possible values: 0 corresponding to an irrelevant true positive, 1 corresponding to a relevant true positive, and 0.5 corresponding to a situation that cannot be verified. Lastly, the importance of evidence (value in $[0, 1]$) is the greater of the importance values of the two hosts connected by the evidence edge.

Two kinds of evidence can be used to construct evidence graphs: primary evidence that is explicit and direct, and secondary evidence that is implicit or circumstantial. In some cases, evidence does not exist or has been destroyed and a subject matter expert may insert an edge corresponding to his expert opinion.

Definition 3. Logical Attack Graph [6, 9]: $A = (N_r, N_p, N_d, E, L, G)$ is a logical attack graph where N_r, N_p and N_d are sets of derivation, primitive and derived fact nodes, $E \subseteq ((N_p \cup N_d) \times N_r) \cup (N_r \times N_d)$, L is a mapping from a node to its label, and $G \subseteq N_d$ is the final goal of an attacker.

Figure 1 shows a logical attack graph. A primitive fact node (box) represents a specific network configuration or vulnerability information corresponding to a host. A derivation node (ellipse) represents a successful application of an interaction rule on input facts, including primitive facts and prior derived facts. The successful interaction results in a derived fact node (diamond), which is satisfied by the input facts.

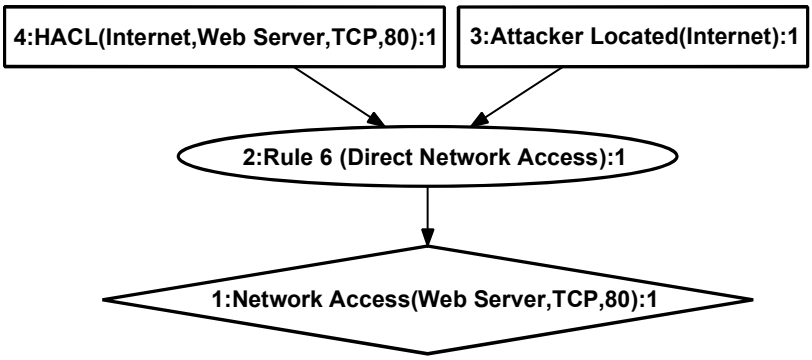


Figure 1. Example attack graph.

Definition 4. Cumulative Probability Function [6]: Given a probabilistic attack graph $A = (N_r, N_p, N_d, E, L, G, p)$, the function $p : N_p \cup N_r \cup N_d \rightarrow [0, 1]$ assigns probabilities to nodes. Exploits (e) are represented as derivation nodes N_r and conditions (c) as primitive facts N_p or derived facts N_d. The cumulative probability function P for e and c of a probabilistic attack graph $P : N_p \cup N_r \cup N_d \rightarrow [0, 1]$ is computed from $p : N_p \cup N_r \cup N_d \rightarrow [0, 1]$ as follows:

- $P(e) = p(e) \times \prod P(c)$ where $e \in N_r$ and c are conditions that include primitive facts and derived facts from a prior step.
- $P(c) = p(c) = 1$ if the condition is a primitive fact that is always satisfied.
- $P(c) = p(c) \times \bigoplus P(e)$ where c is the condition of the derived fact node that is derived from derivation nodes (e).

Definition 5. Sub Logical Attack Graph: $A' = (N'_r, N'_p, N'_d, E', L', G')$ is a sub logical attack graph of a complete logical attack graph $A = (N_r, N_p, N_d, E, L, G)$ if the following conditions hold:

- $N'_r, N'_p, N'_d \subseteq N_r, N_p, N_d$.
- $E' \subseteq E \cap ((N_1, N_2) \in E' \rightarrow N_1, N_2 \in N'_r \cup N'_p \cup N'_d$.
- $G' \subseteq G$, and $L' \subseteq L$.

Definition 6. Similar Nodes: In a logical attack graph $A = (N_r, N_p, N_d, E, L, G)$ or evidence graph $G = (N, E, N - Attr, E - Attr, L, T)$, N_{1d} and N_{2d} are similar if both N_{1d} and $N_{2d} \in A$ satisfy the equalities $N_{1d} \equiv N_{2d}$, $N_{1r} \equiv N_{2r}$ and $N_{1p} \equiv N_{2p}$, represented as $N_{1d} \approx N_{2d}$.

Nodes N_1 and N_2 are similar if they are the same type of hosts $N_1 \wedge N_2 \in G$, the attack status of N_1 is equal to the attack status of N_2, and $E(N_1) \equiv E(N_2)$ where $E(N_1)$ and $E(N_2)$ are the evidence edges whose source or destination host is N_1 and N_2.

Wang and Daniels [13] have suggested normalizing all evidence to five components: (i) ID; (ii) source; (iii) destination; (iv) content; and (v) timestamp. These are used as edges to connect hosts in a time sequence in order to produce an evidence graph.

3. Merging Sub Evidence Graphs

Merging the probabilistic evidence graphs of all the victim hosts and attack evidence in a network provides a global attack description. In order to generate a valid integrated evidence graph, the following guidelines must be followed to ensure that the probabilities of merged host nodes and evidence edges do not violate Definition 2.

- A different node is used to represent each attack on the same host.
- If a victim host is exploited by the same vulnerability in different attacks, the nodes are merged as one node, which is given an increased probability $p'(h) = p(h)_{G1} + p(h)_{G2} - p(h)_{G1} \times p(h)_{G2}$. Note that $p'(h)$ is the increased probability of the merged node, and $p(h)_{G1}$ and $p(h)_{G2}$ are the host probabilities in evidence graphs G_1 and G_2.
- Similar hosts involved in similar attacks are merged together with an increased probability $p'(h) = p(h)_{G1} + p(h)_{G2} - p(h)_{G1} \times p(h)_{G2}$.

When nodes are merged, the corresponding edges that represent the same evidence should also be merged. For evidence probabilities $p(e)_{G1}$ and $p(e)_{G2}$ in evidence graphs G_1 and G_2, the merged evidence probability is increased to $p'(e) = p(e)_{G1} + p(e)_{G2} - p(e)_{G1} \times p(e)_{G2}$ where $p'(e)$ is the new merged evidence edge probability.

Figures 2(a) and 2(b) show two evidence graphs and Figure 2(c) shows their integrated evidence graph. In the two evidence graphs, assume that Host2 is attacked by using the same vulnerability from Host1 and Host3, and assume that Host1 $\approx$ Host3. Host2 in Figures 2(a) and 2(b) can be merged to a single Host2 node with increased probability $p'(h2) = 0.5 + 0.9 - 0.5 \times 0.9 = 0.95$. Also, Host1 and Host3 from the two graphs can be merged as Host with increased probability $p'(h) = p(h1) + p(h3) - p(h1) \times p(h3) = 0.5 + 0.5 - 0.5 \times 0.5 = 0.75$. The merged evidence is, therefore, increased to $p'(e) = p(e1) + p(e3) - p(e1) \times p(e3) = 2 \times 0.5 - 0.5 \times 0.5 = 0.75$.

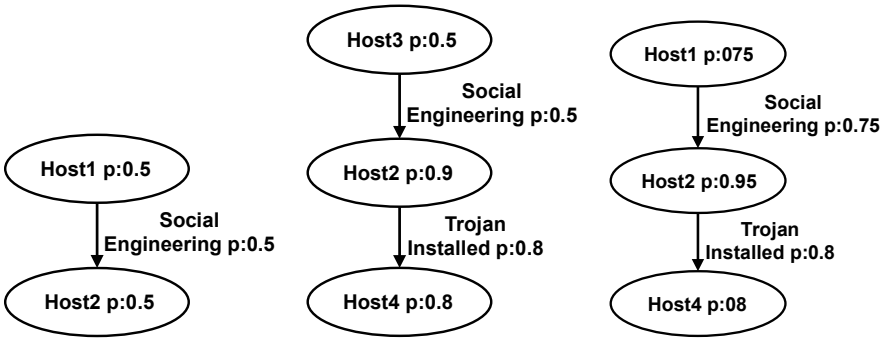


Figure 2. Merging evidence graphs.

If the sub evidence graphs are well constructed with no tainted or missing evidence, then the sub evidence graphs can be merged into an integrated evidence graph. If not, an attack graph generated from the same network is used as a reference to integrate the sub evidence graphs.

3.1 Merging Without an Attack Graph

Algorithm 1 merges two evidence graphs. Starting with the two victim host nodes, the depth-first search algorithm [2] traverses all the host nodes in both evidence graphs and merges them to create an integrated evidence graph. The algorithm uses a coloring scheme to keep track of nodes in the two sub evidence graphs. All the nodes are initialized as being `white`; they are colored `gray` when they are being considered and their children have not yet been fully examined. A host is colored `black` after all its children are fully examined. The merging is complete when all the nodes in the two evidence graphs are colored `black`.

Algorithm 1 takes two sub evidence graphs G_1 and G_2 as input and creates their integrated evidence graph G. The algorithm checks G_1 and G_2 to see if there is an identical host node in G. Note that identical host means the same machine is present in both evidence graphs and the same vulnerability is exploited. If a host node in G is found to be identical to a node in G_1 or G_2, then the probability of the host in G is increased. If there is no identical host in G, then the node and its corresponding evidence from G_1 or G_2 are added to G.

3.2 Merging Using an Attack Graph

Under some circumstances, the constructed evidence graph has nodes that are not connected because of missing evidence edges. In such a case, merging evidence graphs becomes problematic. To address this

Algorithm 1 : Merging Probabilistic Evidence Graphs.
Input: Two evidence graphs $G_i = (N_i, E,_i , N_i - Attr, E_i - Attr, L_i, T_i)$ with victim nodes v_i for $i = 1, 2$.
Output: One evidence graph $G = (N, E, N - Attr, E - Attr, L, T)$.

begin:
1: **for all** nodes $n_1 \in G_1$ and $n_2 \in G_2$ **do**
2: color$[n_1] \leftarrow$ `white`, color$[n_2] \leftarrow$ `white`
3: $\pi[n_1] \leftarrow$ NIL, $\pi[n_2] \leftarrow$ NIL
4: **end for**
5: **for all** nodes $h \in V[G_1] \vee h \in V[G_2]$ **do**
6: **if** color$[h]$ = `white` **then**
7: DFS-VISIT(h)
8: **end if**
9: **end for**
end

DFS-VISIT(h)
1: color$[h] \leftarrow$ `gray`
2: MERGING$(\pi[h], h, G)$
3: **for all** $v \in Adj[h]$ **do**
4: **if** color$[v]$ = `white` **then**
5: $\pi[v] \leftarrow h]$
6: DFS-VISIT(v)
7: **end if**
8: color$(h) \leftarrow$ `black`
9: **end for**
10: **return**

MERGING$(\pi[h], h, G)$
1: **for all** nodes $u \in G$ **do**
2: **if** u is identical to h **then**
3: $p'(u) = p(u) + p(h) - p(u) \times p(h)$, $p'(u.e) = p(u.e) + p(h.e) - p(u.e) \times p(h.e)$
4: **else**
5: add h to G as u', $p(u') = p(h)$
6: **for all** nodes $n \in G$ **do**
7: **if** n is identical to $\pi[h]$ **then**
8: $\pi[u'] \leftarrow n$
9: **else**
10: $\pi[u'] \leftarrow$NIL
11: **end if**
12: **if** $\pi[u'] \neq$NIL **then**
13: $E(\pi[u'], u') \leftarrow E(\pi[h], h)$, $P(E(\pi[u'], u') \leftarrow p(E(\pi[h], h))$
14: **end if**
15: **end for**
16: **end if**
17: **end for**
18: **return**

problem we use an attack graph of the entire enterprise network to fill in the missing evidence by mapping sub evidence graphs to the attack graph to locate the corresponding attack path.

Runtime overhead is a problem when analyzing large-scale attack graphs [3]. It is impractical to map evidence graphs that have poly-

nomial complexity to an attack graph that has an exponential number of nodes V and edges E because the time requirement for a mapping algorithm using a depth-first search is $O(V + E)$ [2]. The following methods are used to reduce attack graph complexity:

- Hosts that have similar vulnerabilities are grouped together. The grouped node is assigned a higher probability $p(h) = p(h_1 \cup h_2) = p(h_1) + p(h_2) - p(h_1) \times p(h_2)$ where h is the grouped machine node, h_1 and h_2 represent Host1 and Host2, respectively, that have similar vulnerabilities [3].

- Attack success probability also represents attack reachability. The hosts with low reachability are removed [7]. We use a value 0.02 as the minimum reachability.

- All paths for which the destination computer does not expand to the desired victim hosts and all hosts that are not involved in the desired attack paths are removed. In addition, if there are multiple exploits at the same host, then the exploits for which the Common Vulnerability Scoring System (CVSS) is smaller than a pre-selected threshold value are omitted.

Algorithm 2 uses an attack graph to merge evidence graphs that have missing evidence. It enhances Algorithm 1 by looking up the missing evidence in attack graph A. The alterations are in the MERGING algorithm, which searches for the missing evidence between a new added node u' and its parent node $\pi[u']$. If the evidence is missing in both G_1 and G_2, then it is searched for in A. Specifically, Lines 6–8 in MERGING search for the evidence edges between $\pi[h]$ and h in G_1 or G_2, which correspond to evidence edges between $\pi[u']$ and u' in G. If these evidence edges are found, they are added as evidence edges between $\pi[u']$ and u' in G. If not, the evidence is missing in both the sub evidence graphs and Lines 15–19 traverse all the derived nodes in attack graph A to find the corresponding nodes of $\pi[u']$ and u' from G, and color them `black`. Line 20 adds the attack paths that are found between the two marked nodes in the attack graph A to the integrated evidence graph G as evidence edges between $\pi[u']$ and u'.

3.3 Complexity Analysis

The complexity analysis uses n and e to represent the numbers of all the nodes and edges in the two evidence graphs. In Algorithm 1, Lines 1–3 have $O(n)$ time because the three lines only go through each node once. Lines 4–6 traverse every node to do a depth-first search (DFS-VISIT),

Algorithm 2 : Integrating Evidence Graphs Using an Attack Graph.
Input: Two evidence graphs $G_i = (N_i, E,_i, N_i - Attr, E_i - Attr, L_i, T_i)$ with victim nodes v_i for $i = 1, 2$. One attack graph $A = N_r, (N_p, E_i, N_i - Attr, E_i - Attr, L_i, T_i)$.
Output: One evidence graph $G = (N, E, N - Attr, E - Attr, L, T)$.

begin:
1: **for all** nodes $n_1 \in G_1$ and $n_2 \in G_2$ **do**
2: color$[n_1] \leftarrow$ `white`, color$[n_2] \leftarrow$ `white`, $\pi[n_1] \leftarrow$ NIL, $\pi[n_2] \leftarrow$ NIL
3: **end for**
4: **for** each node h $\in$ V$[G_1]$ or h $\in$ V$[G_2]$ **do**
5: **if** color$[h]$ = `white` **then**
6: DFS-VISIT(h)
7: **end if**
8: **end for**
end

DFS-VISIT(h)
1: color$[h] \leftarrow$ `gray`, MERGING$(\pi[h], h, G, A)$
2: **for all** $v \in Adj[h]$ **do**
3: **if** color$[v] \equiv$ `white` **then**
4: $\pi[v] \leftarrow h$, DFS-VISIT(v)
5: **end if**
6: **end for**
7: color$[h] \leftarrow$ `black`
8: **return**

MERGING$(\pi[h], h, G, A)$
1: **for all** nodes $u \in G$ **do**
2: **if** u is identical to h **then**
3: $p'(u) = p(u) + p(h) - p(u) \times p(h)$, $p'(u.e) = p(u.e) + p(h.e) - p(u.e) \times p(h.e)$
4: **else**
5: add h to G as u', $p(u') = p(h)$
6: **for all** nodes $n \in G$ **do**
7: **if** n is identical to $\pi[h]$ **then**
8: $\pi[u'] \leftarrow n$
9: **else**
10: $\pi[u'] \leftarrow$ NIL
11: **end if**
12: **if** $\pi[u'] \neq$NIL and $E(\pi[h], h]) \neq$NIL **then**
13: $E(\pi[u'], u') \leftarrow E(\pi[h], h)$, $P(E(\pi[u'], u')) \leftarrow p(E(\pi[h], h))$
14: **else**
15: **for all** derived nodes $m \in A$ **do**
16: **if** m is identical to $\pi[u'] \vee u'$ **then**
17: color$[\pi[u']] \leftarrow$`black`, color$[u'] \leftarrow$`black`
18: **end if**
19: **end for**
20: $E(\pi[u'], u') \leftarrow$ Path between two marked nodes in A
21: **end if**
22: **end for**
23: **end if**
24: **end for**
25: **return**

which has $O(n)$ time. DFS-VISIT calls the MERGING function, which includes two loops that have $O(n^2)$ time. Depth-first search traverses

every edge and node in both evidence graphs, which takes $O(n+e)$ time as proved in [2]. Upon totaling these times, Algorithm 1 has complexity $O(n+n\times(n\times e+n\times n^2)) = O(n^2\times e+n^4)$. For most evidence graphs, e is polynomial to n because there are usually at most three edges between two nodes, so the execution time is polynomial.

Algorithm 2 is similar to Algorithm 1, except that Algorithm 2 has to traverse all the derived nodes in the corresponding attack graph to find the attack steps corresponding to missing evidence in the evidence graphs. This is done by a loop in MERGING (Lines 15–19). Using m to represent the node number in the attack graph, the complexity of Algorithm 2 is $O(n+n\times(n\times e+n\times n^2\times m)) = O(n^2\times e+n^4\times m)$, which is determined by m because e is polynomial to n.

Ou, *et al.* [9] have proved that a logical attack graph for a network with N machines has a size at most $O(N^2)$. However, in a large-scale network with large N, an $O(N^2)$ size can still be a problem. By reducing the number N to only the hosts involved in evidence graphs in the corresponding attack graph, the size of the attack graph can be made smaller. Therefore, the complexity of Algorithm 2 is polynomial instead of exponential because m (equivalent to N) is polynomial.

4. Example Attack Scenario

Figure 3 shows the network used to simulate an attack scenario. In this network, an external firewall controls access from the Internet to the enterprise network, where an Apache Tomcat web server using Firefox 3.6.17 hosts a website that allows access to Internet users via port 8080. The internal firewall controls access to a MySQL database server that also functions as a file server. The web server can access the database server via the default port 3306 and the file server through the NFS protocol. Workstations with Microsoft Internet Explorer 6 (IE6) installed have direct access to the internal database server.

The attacker's objective is to gain access to database tables stored on the database server and to compromise the file server. The attack plan involves: (i) an SQL injection attack that exploits a Java servlet on the web server; and (ii) direct access by compromising a workstation.

The Java servlet on the web server does not sanitize inputs; it is vulnerable to: `theStatement.executeQuery("SELECT * FROM profiles WHERE name='Alice' AND password=' "+passWord+" ' ");`, corresponding to CWE89 in NVD [5]. The workstations run Windows XP SP3 with IE6, which has the vulnerability CVE-2009-1918 [8]. This vulnerability allows an attacker to use social engineering to enable his machine to control the targeted workstation. The file server attack plan uses a

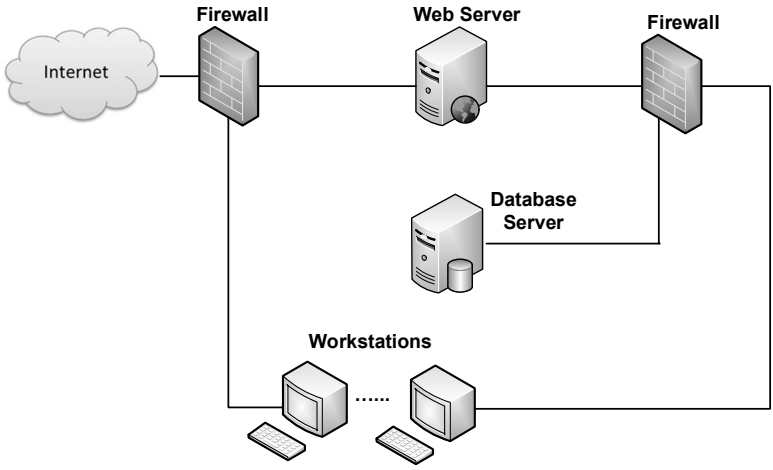


Figure 3. Experimental network.

Table 1. Attack role configuration.

System	Role	Vulnerability
129.174.128.148	Attacker	
Workstation	Stepping Stone	CVE-2009-1918
Web Server	Stepping Stone/Affiliated	CWE89/CVE-2011-236
Database Server/File Server	Victim	CVE-2003-0252

vulnerability in NFS service daemons (CVE-2003-0252) to compromise the file server. The web server, which is a stepping stone to attack the file server, can be compromised by exploiting vulnerability CVE-2011-2365 in Firefox 3.6.17. Table 1 lists the attack roles and vulnerabilities.

4.1 Merging Graphs Using Algorithm 1

The evidence from the attack scenario is modeled as a vector (ID, source, destination, content, timestamp) and appears in the evidence graphs in Figures 4(a) and 4(b). In both graphs, solid edges represent primary evidence and secondary evidence, which have coefficients of 1 and 0.8, respectively. Dotted edges represent expert knowledge with a coefficient of 0.5. The probabilities for the evidence edges and hosts are calculated using Definition 2.

Executing Algorithm 1 yields the integrated evidence graph in Figure 4(c). Because the workstations in the two sub evidence graphs are involved in the same attack, they are merged to a single node with an increased probability $p'(h) = p(h.G_1) + p(h.G_2) - p(h.G_1) \times p(h.G_2) =$

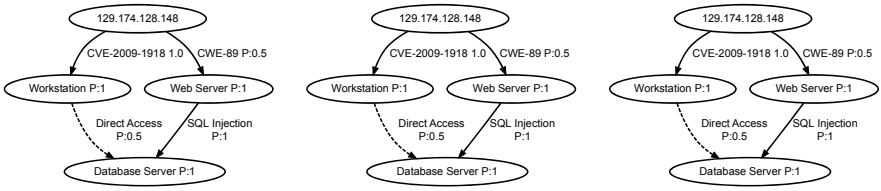


Figure 4. Evidence graphs for the experimental network attack scenario.

$1 + 1 - 1 \times 1 = 1$. Note that $p(h.G_1)$ and $p(h.G_2)$ are the probabilities of $p(h)$ in evidence graphs G_1 and G_2, respectively.

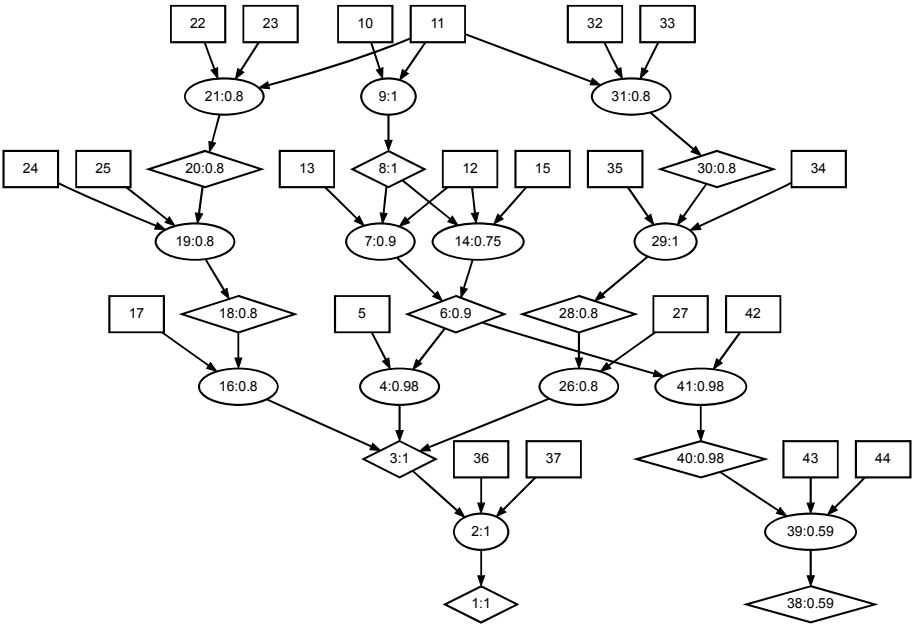


Figure 5. Attack graph with two workstations.

4.2 Merging Graphs Using Algorithm 2

Figure 5 shows the attack graph with the vulnerability information in Table 1. The graph shows two attack paths on the left and right sides. Merging the two workstations yields Figure 6, which has less computational complexity than Figure 5. The grouped host nodes have updated probabilities of:

$$p'(21) = p(21) + p(31) - p(21) \times p(31) = 0.8 + 0.8 - 0.8 \times 0.8 = 0.96$$
$$p'(20) = p(20) + p(30) - p(20) \times p(30) = 0.8 + 0.8 - 0.8 \times 0.8 = 0.96$$

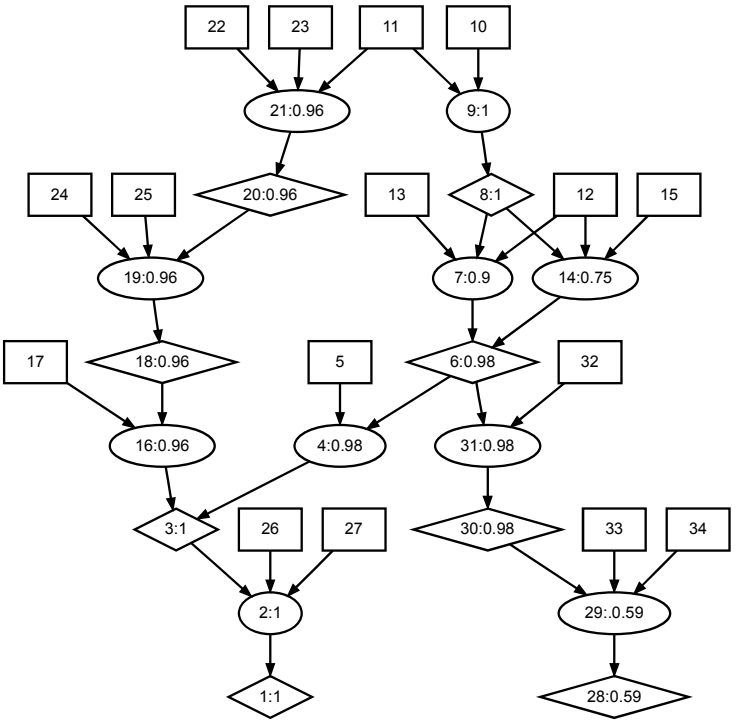


Figure 6. Attach graph with grouped workstations.

$$p'(19) = p(19) + p(29) - p(19) \times p(29) = 0.8 + 0.8 - 0.8 \times 0.8 = 0.96$$
$$p'(18) = p(18) + p(28) - p(18) \times p(28) = 0.8 + 0.8 - 0.8 \times 0.8 = 0.96$$

Algorithm 2 uses the compact attack graph in Figure 6 to integrate the two evidence graphs. In order to simulate missing evidence in the evidence graphs, we assume that the evidence from the web server to the file server in the graph in Figure 4(b) has not been found. The corresponding integrated evidence graph is shown in Figure 7. In this graph, the dashed edge is merged from the two evidence graphs and the attack path between the web server and the file server is the corresponding attack path (between node 6 and node 28) from the attack graph. This new added attack path between the web server and the file server indicates that the web server has been used as a stepping stone to compromise the file server using vulnerability CVE-2003-0252.

5. Conclusions

Two algorithms are presented for merging probabilistic sub evidence graphs to create an integrated probabilistic evidence graph. The first algorithm merges two probabilistic evidence graphs without using an

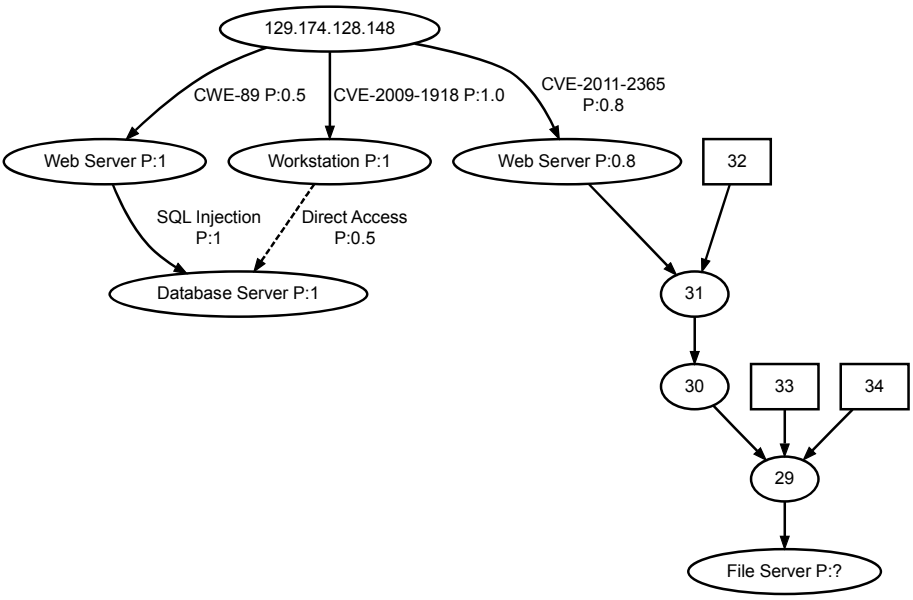


Figure 7. Integrated evidence graph.

attack graph of infrastructure vulnerabilities. The second algorithm, which merges multiple evidence graphs using an attack graph, is used in situations where evidence has been tampered with or is missing. Because the second algorithm can have a lengthy execution time for an attack graph with high complexity, certain approximations can be applied to shrink attack graphs to reduce the execution time.

Note that this paper is not subject to copyright in the United States. Commercial products are identified in the paper in order to present and evaluate certain procedures. The identification of the products does not imply a recommendation or endorsement by the National Institute of Standards and Technology or the U.S. Government, nor does it imply that the identified products are necessarily the best available products for the task.

References

[1] P. Ammann, D. Wijesekera and S. Kaushik, Scalable, graph-based network vulnerability analysis, *Proceedings of the Ninth ACM Conference on Computer and Communications Security*, pp. 217–224, 2002.

[2] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2009.

[3] J. Homer, A. Varikuti, X. Ou and M. McQueen, Improving attack graph visualization through data reduction and attack grouping, *Proceedings of the Fifth International Workshop on Visualization for Cyber Security*, pp. 68–79, 2008.

[4] K. Ingols, R. Lippmann and K. Piwowarski, Practical attack graph generation for network defense, *Proceedings of the Twenty-Second Annual Computer Security Applications Conference*, pp. 121–130, 2006.

[5] S. Jha, O. Sheyner and J. Wing, Two formal analyses of attack graphs, *Proceedings of the Fifteenth Computer Security Foundations Workshop*, p. 49, 2002.

[6] C. Liu, A. Singhal and D. Wijesekera, Mapping evidence graphs to attack graphs, *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 121–126, 2012.

[7] V. Mehta, C. Bartzis, H. Zhu, E. Clarke and J. Wing, Ranking attack graphs, *Proceedings of the Ninth International Conference on Recent Advances in Intrusion Detection*, pp. 127–144, 2006.

[8] National Institute of Standards and Technology, National Vulnerability Database, Version 2.2, Gaithersburg, Maryland (`nvd.nist.gov`).

[9] X. Ou, W. Boyer and M. McQueen, A scalable approach to attack graph generation, *Proceedings of the Thirteenth ACM Conference on Computer and Communications Security*, pp. 336–345, 2006.

[10] O. Sheyner, J. Haines, S. Jha, R. Lippmann and J. Wing, Automated generation and analysis of attack graphs, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 273–284, 2002.

[11] A. Singhal and X. Ou, Security Risk Analysis of Enterprise Networks using Probabilistic Attack Graphs, NIST Interagency Report 7788, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[12] L. Wang, T. Islam, T. Long, A. Singhal and S. Jajodia, An attack graph based probabilistic security metric, *Proceedings of the Twenty-Second Annual IFIP WG 11.3 Conference on Data and Applications Security*, pp. 283–296, 2008.

[13] W. Wang and T. Daniels, A graph based approach toward network forensic analysis, *ACM Transactions on Information and Systems Security*, vol. 12(1), article no. 4, 2008.

Chapter 17

A GENERIC BAYESIAN BELIEF MODEL FOR SIMILAR CYBER CRIMES

Hayson Tse, Kam-Pui Chow and Michael Kwan

Abstract Bayesian belief network models designed for specific cyber crimes can be used to quickly collect and identify suspicious data that warrants further investigation. While Bayesian belief models tailored to individual cases exist, there has been no consideration of generalized case modeling. This paper examines the generalizability of two case-specific Bayesian belief networks for use in similar cases. Although the results are not conclusive, the changes in the degrees of belief support the hypothesis that generic Bayesian network models can enhance investigations of similar cyber crimes.

Keywords: Bayesian networks, DDoS attacks, BitTorrent file sharing

1. Introduction

One of two alternative strategies is typically employed to collect digital evidence during an investigation [5]. The first is to use staff with limited forensic training and seize everything. The second is to use skilled experts to perform selective acquisition. Given the resource constraints and expanding file storage capacities, it is not always feasible to retrieve all the evidence and to conduct a thorough analysis of all the digital traces. Indeed, there is no option for first responders but to make on-site acquisition decisions.

Sullivan and Delaney [13] recommend that investigators should analyze incomplete information (prior probability distributions), adjust their opinions (conditional probabilities) based on experience (observed information), and incorporate all the relevant information in an analytical process that reflects the extrapolation of experience. Bayesian reasoning supports these recommendations. It provides an investigator with a numerical procedure to revise beliefs based on expert knowledge

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 243–255, 2013.

and any new evidence that is collected. The investigator computes the probability of an evidentiary item under a given hypothesis and evaluates the likelihood that the evidentiary item is conclusive based on the hypothesis.

This paper examines the feasibility of designing a generic Bayesian network model that applies to similar cyber crimes. Two Bayesian network models are examined, one constructed for a distributed denial-of-service (DDoS) attack case that is used in a denial-of-service (DoS) case, and the other constructed for a BitTorrent file sharing case that is used for an eMule file sharing case. Hypothesis testing of the degrees of belief supports the notion that a generic per-case model can be applied to similar cases.

2. Forensic Case Assessment and Triage

Backlog in digital forensic laboratories is common. In 2009, the United Kingdom Association of Chief Police Officers described the backlog in analyzing seized data "as one of the biggest e-crime problems" [2]. Experience has shown that an increase in the number of staff alone does not reduce backlogs.

Triage is useful in situations involving limited resources. In a medical emergency, if there are insufficient resources to treat all the victims, the casualties are sorted and prioritized. Treatments are targeted towards victims who can benefit the most. Those who are beyond help or do not need treatment urgently are not treated or are treated later. The medical community has adopted methods and protocols for determining how to prioritize and treat victims during triage.

Similarly, triage should be performed when there is insufficient time to comprehensively analyze digital evidence at a crime scene. When performing triage, a first responder would attempt to quickly identify suspicious data that warrants further investigation and eliminate data that is not relevant. Sebastian and Gomez [5] have shown that laboratory workloads can be reduced by performing triage using automated tools. Additionally, Rogers, *et al.* [12] have proposed a model for on-site identification, analysis and interpretation of digital evidence without having to acquire a complete forensic image.

Bayesian belief networks have been used to determine if investigations are worth undertaking [9]. Overill and Silomon [10] use the term "digital metaforensics" to quantify the investigation of digital crime cases. They argue that a preliminary filtering or pre-screening phase could help rank the probable order of evidential strength. Overill, *et al.* [9] emphasize that it is the duty of digital forensic practitioners to retrieve digital

traces to prove or refute alleged computer acts. They maintain that, given the resource constraints, it is not always feasible or necessary to retrieve all the related digital traces and to conduct a thorough digital forensic analysis.

Overill, *et al.* [9] and Cohen [3] have specified cost-effectiveness metrics for conducting cost-benefit analyses of forensic investigations, including cost-efficiency [9] and return-on-investment studies [3]. In particular, Overill, *et al.* [9] have proposed a two-phase schema for performing digital forensic examinations at minimal cost. The first phase of the schema is pre-processing, which is the detection of digital traces. Pre-processing includes enumerating the set of traces that are expected to be present in a seized computer; the enumeration is based on the type of computer crime that is suspected of having been committed. The second phase of the schema is the analysis of the traces, for which a Bayesian belief network is used.

3. Bayesian Networks

A Bayesian network offers several advantages to digital forensic practitioners. After a Bayesian network has been constructed, it is easy to understand and apply, which can speed up an investigation. Also, a Bayesian network ensures that all the available information is considered. This can prevent human error in overlooking small, but nevertheless, vital factors. A Bayesian network can also be used to make inferences and find patterns that may be too complicated for human practitioners.

Bayesian networks are graphical structures that represent probabilistic relationships between a number of variables and support probabilistic inference with the variables [8]. According to Heckerman, *et al.* [6], the Bayesian probability of an event x is a person's "degree of belief" in the event. Data is used to strengthen, update or weaken the belief.

The nodes in a Bayesian network represent a set of random variables $X = \{X_1, \ldots X_i, \ldots, X_n\}$ defined by two components:

- **Qualitative Component:** A directed acyclic graph (DAG) in which each node represents a variable in the model and each edge linking two nodes (i.e., variables) indicates a statistical dependence between the nodes.

- **Quantitative Component:** A conditional probability distribution $p(x_i|pa(x_i))$ for each variable X_i, $i = 1, \ldots, n$, given its parents in the graph $pa(x_i)$.

A DAG represents a set of conditional independence statements regarding the nodes. Each node is annotated with a conditional distri-

bution $P(X_i \mid Parents(X_i))$. Let $Parents(V)$ be the set of parents of a variable V in a DAG G and let $Descendants(V)$ be the set of the descendants of variable V. Then, the DAG G expresses the independence statements: for all variables V in G: $I(V, Parents(V), NonDescendants(V))$ [11], which means that every variable is conditionally independent of its non-descendants given its parents. In other words, given the direct causes of a variable, the beliefs in the variable are not influenced by any other variable except possibly by its effects.

A Bayesian network yields a complete joint probability distribution for all possible combinations of variables over X given by:

$$P(X_1, \ldots, X_i) = \prod_{i=1}^{n} P(X_i | Parents(X_i)).$$

Constructing a Bayesian network involves three tasks. First, the variables and their initial probabilities are identified. Second, the relationships between the variables are identified and expressed in a graphical structure; this is usually done by domain experts and is comparable to knowledge engineering. Third, the probabilities required for the quantitative analysis are assigned. The probabilities are generally obtained from statistical data, the research literature and human experts. Automated construction of a Bayesian network is feasible when there is an adequate amount of unbiased data.

4. Generic Bayesian Network Crime Model

At the start of an investigation, a Bayesian network model for the crime must be available. The first responders at the crime scene use the Bayesian network to quickly identify relevant information.

Figure 1 shows an example Bayesian network for a DDoS attack created using the SamIam 3.0 modeling tool [1]. In a Bayesian network, "No" indicates that the event described by the corresponding variable did not occur; "Yes" indicates that the event occurred; and "Uncertain" indicates that there is doubt if the event occurred or not. All the available evidence is entered into the network by selecting a state for each variable and, if necessary, entering a probability distribution.

During an investigation, if a first responder finds evidence associated with all the variables in the Bayesian network, then all the nodes in the network would be set to "Yes." Upon propagating the probability computations in the example Bayesian network, the probabilities that the hypotheses H_1, H_2 and H are "Yes" become 99.73%, 100.00% and 93.26%, respectively. The result is shown in Figure 2.

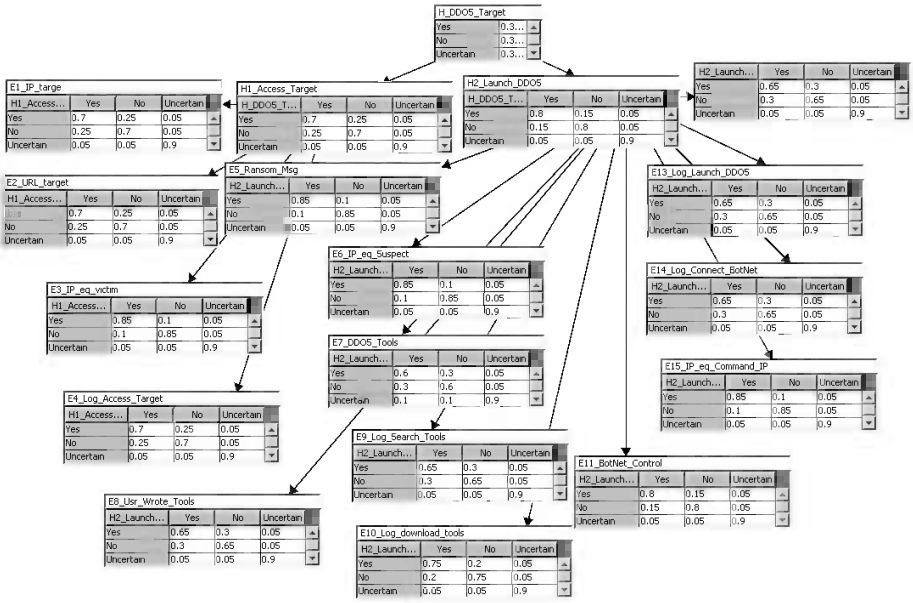


Figure 1. Bayesian network for a DDoS attack.

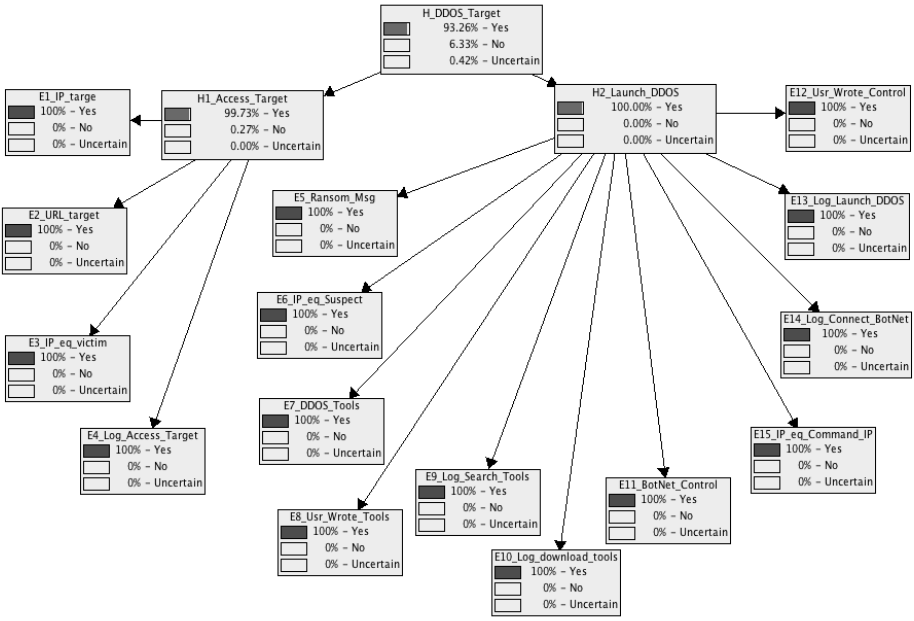


Figure 2. Bayesian network for a DDoS attack (variables set to "Yes").

Of course, if the first responder finds evidence associated with some of the variables, then only the associated nodes in the network would be set to "Yes." The values of the hypotheses H_1, H_2 and H would change accordingly.

Bayesian networks cannot be expected to give perfect answers. A Bayesian network is a simplification of a complicated situation, operating on information that is uncertain in the first place. Furthermore, a network only gives the likelihood of occurrence of a particular event (e.g., hypothesis).

Generic Bayesian network models of similar types of cyber crimes can be used when conducting triage at crime scenes. Experienced Hong Kong criminal investigators have developed Bayesian networks to represent the relationships between events and hypotheses in a DDoS attack case and in a BitTorrent file sharing case. The two networks are used to explore the feasibility of employing generic Bayesian networks in criminal investigations. In the case of the first network, we assume that a DoS attack is an example of a DDoS attack in that a DoS attack is from a single source rather than from multiple sources. The second network explores the generalizability of the BitTorrent Bayesian network to the eMule peer-to-peer (P2P) file sharing network.

In the case of the Bayesian network for a DDoS attack, the main hypothesis H is:

- H**:** Seized computer was used to launch a DDoS attack against a target computer.

The sub-hypotheses are:

- H_1**:** Seized computer was used to access the target computer.
- H_2**:** Seized computer was used to launch a DDoS attack.

The evidence supporting H_1 is:

- E_1**:** IP address of the target computer was found on the seized computer.
- E_2**:** URL of the target computer was found on the seized computer.
- E_3**:** IP address of the target computer matched the accessed IP address (revealed by the ISP) at the material time.
- E_4**:** Log file records were found on the seized computer indicating that the target computer was accessed at the material time.

The evidence supporting H_2 is:

- E_5**:** Ransom messages were found on the seized computer for extorting money (or other benefits) from the victim.
- E_6**:** IP address of the seized computer matched the attacking IP address (revealed by the ISP) at the material time.
- E_7**:** DDoS tools were found on the seized computer.
- E_8**:** Evidentiary data was found on the seized computer indicating that the computer user wrote the DDoS tools.

- E_9: Log file records were found on the seized computer indicating that the computer user searched for DDoS tools on the Internet.
- E_{10}: Log file records were found on the seized computer indicating that the computer user downloaded DDoS tools from the Internet.
- E_{11}: The command and control program of a botnet was found on the seized computer.
- E_{12}: Evidentiary data was found on the seized computer indicating that the computer user wrote the command and control program of a botnet.
- E_{13}: Log file records were found on the seized computer indicating that the computer was used to launch a DDoS attack against the target computer via a botnet.
- E_{14}: Log file records were found on the seized computer indicating that the computer was connected to a botnet.
- E_{15}: IP address of the seized computer matched the botnet command and control IP address (revealed from the attacking bots) at the material time.

Our first evaluation assumes that a DoS attack is an example of a DDoS attack. The analysis uses evidence from an actual DoS attack case [4].

The case involved an individual who, on August 12, 2011 and August 13, 2011, launched DoS attacks on the Hong Kong Exchange website for 390 seconds and 70 seconds, respectively. On the suspect's computer, law enforcement authorities found a UDP flooder program, Internet connection logs and screen prints of the websites being attacked.

Based on the evidence, the values of E_1 to E_8 are set to "No" and E_9 to E_{15} are set to "Yes." The probabilities of H_1, H_2 and H given the evidence variables compute to 0.93%, 86.07% and 57.11%, respectively, as shown in Figure 3. Although the values are well below 100%, the suspect was convicted after a trial. At the trial, the suspect admitted that he flooded the website and made records of the attacks for educational purposes.

Our second evaluation uses a Bayesian network model created to explore the evidence observed by crime investigators in a Hong Kong criminal case regarding illegal file sharing using BitTorrent [7, 15]. Kwan, *et al.* [7] established the probability distributions of the hypotheses and evidentiary variables in the case. This enabled the quantification of the evidentiary strengths of the various hypotheses. Tse, *et al.* [14] revised the model and examined two methods for determining whether or not the Bayesian network could be refined. This analysis uses the model established by Tse, *et al.* [14], which is shown in Figure 4. The model incorporates the following hypotheses:

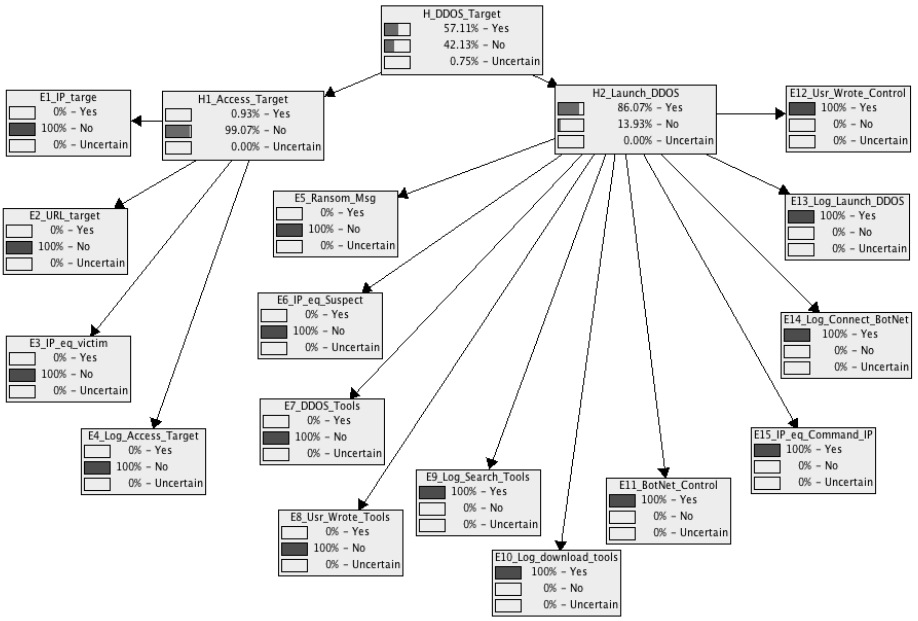


Figure 3. Bayesian network for the DoS attack (based on case evidence).

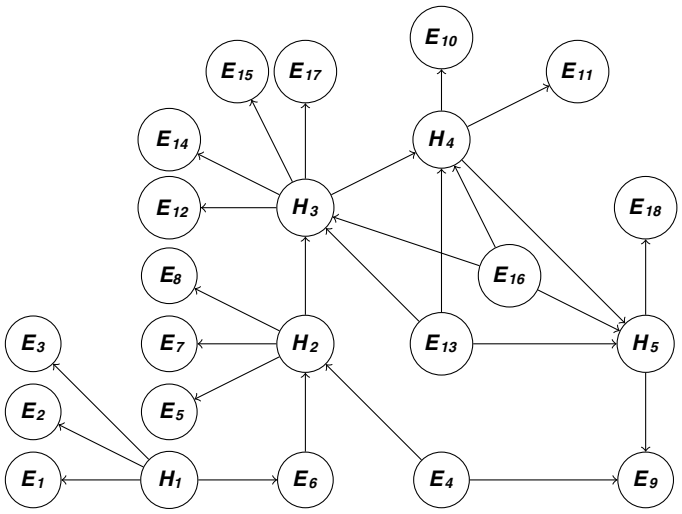


Figure 4. Bayesian network for the BitTorrent case.

- H: Seized computer was used by the seeder to share the pirated file on the BitTorrent network.
- H_1: Pirated file (destination file) was copied from the seized optical disk (source file) to the seized computer.
- H_2: Torrent was created from the pirated file.
- H_3: Torrent was sent to a newsgroup for publishing.
- H_4: Torrent was activated causing the seized computer to connect to the tracker server.
- H_5: Connection between the seized computer and the tracker was maintained.

The evidentiary variables in the model are:

- E_1: Modification time of the destination file was identical to that of the source file.
- E_2: Creation time of the destination file was after its own modification time.
- E_3: Hash value of the destination file matched that of the source file.
- E_4: BitTorrent client software was installed on the seized computer.
- E_5: File link for the pirated file (shared file) was created.
- E_6: Pirated file existed on the hard disk of the seized computer.
- E_7: Torrent creation record was found.
- E_8: Torrent existed on the hard disk of the seized computer.
- E_9: Peer connection information was found on the seized computer.
- E_{10}: Tracker server login record was found.
- E_{11}: Torrent activation time was corroborated by its MAC time and link file.
- E_{12}: Internet history record of the torrent publishing website was found.
- E_{13}: Internet connection was available.
- E_{14}: Cookie of the website of the newsgroup was found.
- E_{15}: URL of the website was stored in the web browser.
- E_{16}: Web browser software was available.
- E_{17}: Internet cache record regarding the publishing of the torrent was found.
- E_{18}: Internet history record regarding the tracker server connection was found.

Figure 5 shows the revised BitTorrent Bayesian network when all the evidentiary variables are set to "Yes." The resulting probabilities for the hypotheses being "Yes" are 97.67% for H_1, 99.64% for H_2, 99.86% for H_3, 98.94% for H_4 and 98.48% for H_5.

In a P2P file sharing network, each computer in the network acts as a client or server. Individuals who want files and individuals who have files are all connected in the network. In the case of the eMule P2P network, the hashes of shared files are maintained in hash lists at eMule servers. Individuals search the servers for files of interest and are presented with the filenames and unique hash identifiers. The individuals then query

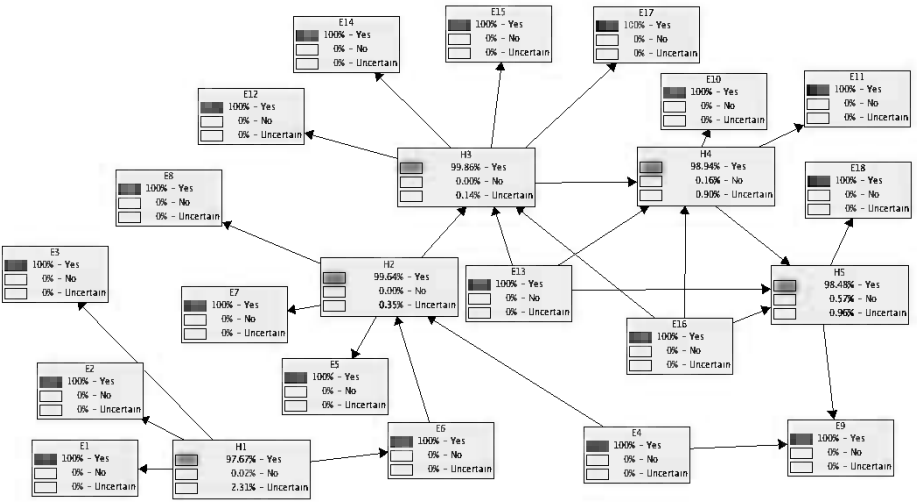


Figure 5. Revised BitTorrent Bayesian network (variables set to "Yes").

the servers for clients that have files that match the hash identifiers. The servers return sets of IP addresses with the locations of the clients. Chunks of the files are then swapped until the individuals who want the files have complete copies of the files.

If A has a complete version of a file that is of interest to B, C, D and E, then a P2P system would enable A to feed B with the first part (say one-quarter) of the file, feed C with the second quarter, D with the third quarter and E with the fourth quarter. Then, B, C, D and E would exchange what they have received from A until all four have the complete file. If A were to cut the connection after it distributed the four chunks, then B, C, D and E could still distribute the file amongst themselves. However, if A were to cut the connection before it distributed all four chunks, then B, C, D and E could distribute what they received amongst themselves, but they would be unable to obtain the complete file until another client joins the network with the complete file.

There are some differences between the BitTorrent and eMule P2P networks. In a BitTorrent network, there is no centralized location for a file or a hash list that is searched to locate files. Instead, users must receive or locate a file on an indexing website and download a file tracker (`.torrent` file). All the users who wish to share the complete file use the tracker to create the P2P network for the file. However, both BitTorrent and eMule need a central location to exchange information regarding the file identity and the IP addresses of users. The concepts of creating

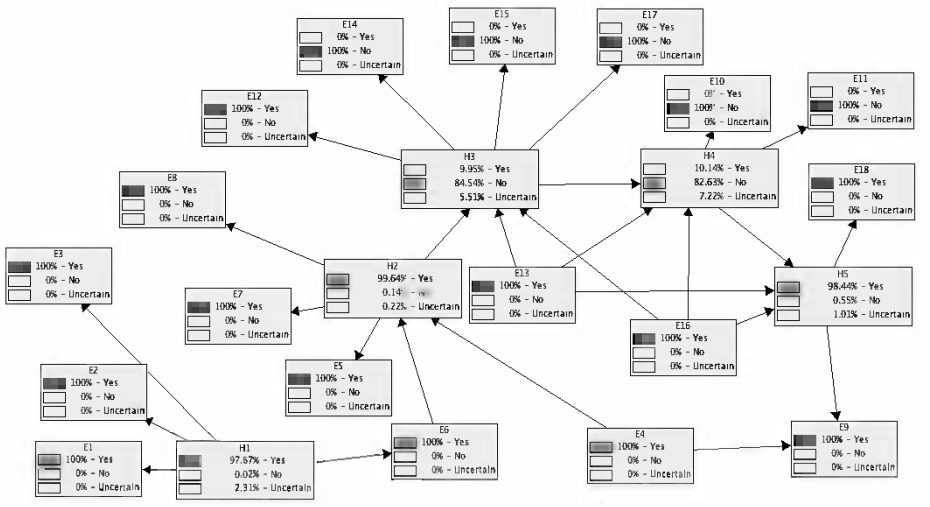


Figure 6. Revised BitTorrent Bayesian network (based on the eMule evidence set).

and using a hash list in eMule are equivalent to creating and using a `.torrent` file in BitTorrent.

For the purpose of the second analysis, in order to use the BitTorrent Bayesian network for an eMule case, the evidentiary items E_{10}, E_{11}, E_{14}, E_{15} and E_{17} are set to "No" while the remaining evidentiary items are set to "Yes." Figure 6 shows the updated Bayesian network. The probabilities that the hypotheses are "Yes" are: 97.67% for H_1, 99.64% for H_2, 9.95% for H_3, 10.14% for H_4 and 98.44% for H_5.

Although the changes to the degrees of belief in the two analyses are not conclusive, they provide support for our hypothesis that a generic Bayesian network model can be used quite effectively to analyze similar cyber crimes.

5. Conclusions

Generic Bayesian network models can be used to improve the quality of cyber crime investigations while reducing the effort and cost. The results of the evaluation of two Bayesian network models, one constructed for a DDoS attack case that was used in a DoS case, and the other constructed for a BitTorrent file sharing case that was used for an eMule file sharing case, support the notion that a generic case-specific model can be applied to similar cases.

Our future research will explore how a generic Bayesian network can be created to accommodate a larger number of similar cyber crimes. This network should be constructed carefully because, as the number of causal

nodes increases, the sizes of the probability matrices of the nodes grow exponentially. Our research plans also involve designing a dedicated user interface with abstraction support to enable law enforcement officers to interact with Bayesian networks in a flexible and intuitive manner.

References

[1] Automated Reasoning Group, SamIam, University of California at Los Angeles, Los Angeles, California (`reasoning.cs.ucla.edu/samiam`), 2010.

[2] R. Blincoe, Police sitting on forensic backlog risk, says top e-cop, *The Register* (`www.theregister.co.uk/2009/11/13/police_forensics_tool`), November 13, 2009.

[3] F. Cohen, Two models of digital forensic examination, *Proceedings of the Fourth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 42–53, 2009.

[4] District Court of the Hong Kong Special Administrative Region, HKSAR against Man-Fai Tse, Criminal Case No. 1318 of 2011, Hong Kong, China, 2012.

[5] L. Gomez, Triage in-Lab: Case backlog reduction with forensic digital profiling, *Proceedings of the Argentine Conference on Informatics and Argentine Symposium on Computing and Law*, pp. 217–225, 2012.

[6] D. Heckerman, A Tutorial on Learning with Bayesian Networks, Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Microsoft, Redmond, Washington, 1996.

[7] M. Kwan, K. Chow, F. Law and P. Lai, Reasoning about evidence using Bayesian networks, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 275–289, 2008.

[8] R. Neapolitan, *Learning Bayesian Networks*, Prentice-Hall, Upper Saddle River, New Jersey, 2003.

[9] R. Overill, M. Kwan, K. Chow, P. Lai and F. Law, A cost-effective model for digital forensic investigations, in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 231–240, 2009.

[10] R. Overill and J. Silomon, Digital meta-forensics: Quantifying the investigation, *Proceedings of the Fourth International Conference on Cybercrime Forensics Education and Training*, 2010.

[11] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Francisco, California, 1997.

[12] M. Rogers, J. Goldman, R. Mislan, T. Wedge and S. Debrota, Computer Forensics Field Triage Process Model, *Journal of Digital Forensics, Security and Law*, vol. 1(2), pp. 19–37, 2006.

[13] R. Sullivan and H. Delaney, Criminal investigations – A decision-making process, *Journal of Police Science and Administration*, vol. 10(3), pp. 335–343, 1982.

[14] H. Tse, K. Chow and M. Kwan, Reasoning about evidence using Bayesian networks, in *Advances in Digital Forensics VIII*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 99–113, 2012.

[15] Tuen Mun Magistrates Court of the Hong Kong Special Administrative Region, HKSAR against Nai-Ming Chan, Criminal Case No. 1268 of 2005, Hong Kong, China, 2005.

[16] Y. Xiang and Z. Li, An analytical model for DDoS attacks and defense, *Proceedings of the International Multi-Conference on Computing in the Global Information Technology*, p. 66, 2006.

Chapter 18

AN EMPIRICAL STUDY PROFILING INTERNET PIRATES

Pierre Lai, Kam-Pui Chow, Xiao-Xi Fan and Vivien Chan

Abstract Internet piracy has become a serious problem due to the expansion of network capacity and the availability of powerful hardware. To combat this problem, industry and law enforcement need a better understanding of the behavioral characteristics of Internet pirates. This paper describes a new conceptual framework for profiling Internet pirates. Also, it presents a taxonomy based on a survey of 114 Internet pirates. The taxonomy, which includes six types of downloaders and six types of file sharers with different behavioral characteristics, provides useful insights to forensic scientists and practitioners who are focused on combating Internet piracy.

Keywords: Internet pirates, criminal profiling, behavioral characteristics

1. Introduction

Internet piracy is the act of illegally copying or distributing copyrighted digital files using the Internet [7]. Over the years, industry and law enforcement agencies have conducted major operations to stop these illegal activities. However, their efforts appear to be unable to reduce Internet piracy. It is estimated that at least 23.76% of the world's Internet bandwidth is devoted to the transfer of infringing, non-pornographic content [6].

Rogers [13] has noted that criminal profiling can benefit investigations of cyber crimes such as Internet piracy because it helps develop effective investigative and media search strategies, and helps reduce the number of possible suspects. Turvey [14] has demonstrated how behavioral evidence of cyber crimes can be analyzed and used to profile offenders. While it is promising to use profiling techniques to understand the behaviors and personal characteristics of Internet pirates, existing Internet

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 257–272, 2013.

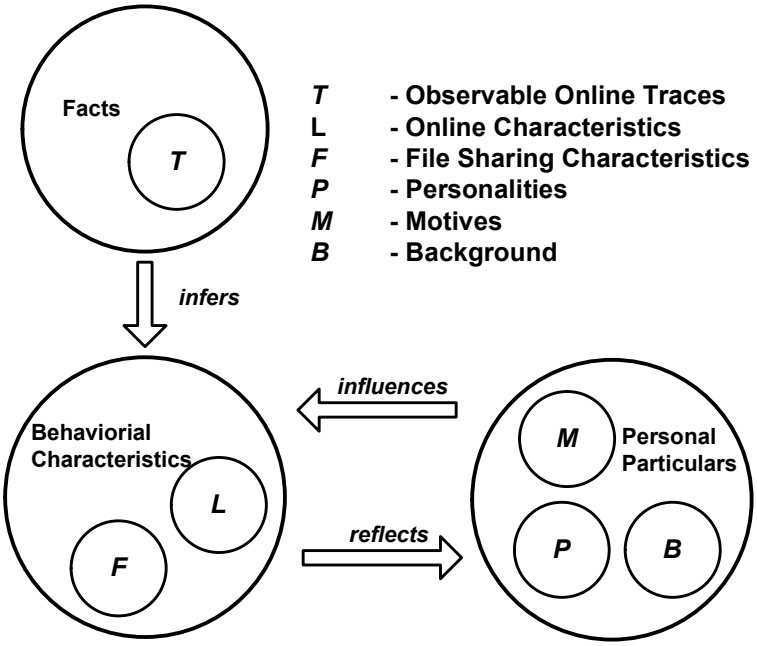


Figure 1. Conceptual framework for profiling Internet pirates.

piracy research (e.g., [7, 8]) has not attempted to relate the behaviors of Internet pirates to their personal characteristics.

This paper describes a conceptual framework for profiling Internet pirates along with a taxonomy of Internet pirates developed using an empirical study based on the framework. The taxonomy is constructed using data gleaned from an extensive survey of individuals involved in Internet piracy related activities. It includes the observed behavioral patterns and associated personal characteristics of twelve types of downloaders and file sharers.

2. Profiling Internet Pirates

Because cases of Internet piracy occur in the cyber world, the characteristics and inferences used in the conventional profiling process need to be redefined. In particular, crime scene evidence, which offers the principal clues in a traditional criminal investigation, is not well-defined in the context of Internet piracy.

The conceptual framework for profiling Internet pirates is shown in Figure 1. The framework incorporates three categories: (i) facts (observable online traces); (ii) behavioral characteristics (online characteristics

Table 1. Category and variable description.

Category	**Variable**	**Description**
Facts	Observable Online Traces ($\mathcal{T}$)	Set of traces observable on the Internet, including visible facts and data that exist online and are associated with a file sharing act. Examples are the name of the user who shared a file and the timestamp when the file was uploaded.
Behavioral Characteristics	Online Characteristics ($\mathcal{L}$)	Set of attributes describing the online habits or usual practices of an individual. Examples are the amount of time a person spends online each day, the period when he or she is most active, and what he or she usually does online.
	File Sharing Characteristics ($\mathcal{F}$)	Set of attributes associated with the file sharing activities of an individual. Examples are how often a person uploads files and the locations where the files are usually published.
Personal Particulars ($\mathcal{PP}$)	Personality ($\mathcal{P}$)	Set of internal traits of an individual. These traits affect how a person feels, thinks or responds to external factors. Examples are if a person is positive and outgoing or quiet and introverted.
	Motive ($\mathcal{M}$)	Set of reasons why a person is engaged in a certain act. Examples are to enjoy the resources for free and to obtain the resources with minimal effort.
	Background ($\mathcal{B}$)	Set of general informational items about a person. In this study, the term "background" takes on a broader meaning. It may refer to any general information about a person. Hobbies and interests and personal experience are also regarded as background information. In our empirical study, a number of general background items were selected to be included in the profiling process. Examples are if a person lives alone, gender, highest academic qualification, occupation and computer experience.

and file sharing characteristics); and (iii) personal particulars (personality, motive and background). Table 1 summarizes the categories and variables used in the framework.

The conceptual framework incorporates three types of relationships:

- **Infer:** This relationship is a direct deduction based on an observable online trace. For example, if a post was published at 1 a.m. on December 31, 2010, it could be inferred that the person who published the post was online around 1 a.m. on that day. The observable online trace is "the timestamp of the post" and the inferred characteristic is "the person was online around the time corresponding to the timestamp."

- **Influence:** This relationship is based on two facts: (i) an individual's personal experience and living environment (both play an important role in behavioral development); and (ii) personality, which includes behavioral tendencies consistent over time that affect an individual's actions [3].

- **Reflect:** Online characteristics and file sharing characteristics are behavioral characteristics that differentiate the behaviors of different people. This relationship is the inverse of influence because an individual's personality, background and motive influence the individual's behavior, and the associated behavior can reveal the individual's personality characteristics.

3. Empirical Study of Internet Pirates

Most empirical criminal profiling studies use data gathered from convicted offenders [10, 14]. However, in Hong Kong, the one and only Internet piracy conviction was in the 2005 HKSAR v. Chan Nai Ming case [5]. Due to the single-case pool, the typical convicted offender approach cannot be used.

Our study collected data on Internet pirates by surveying individuals who have shared copyrighted materials with the aid of Chinese-speaking online public forums. This is considered an acceptable method of data collection for two reasons. The first is that in the Chan Nai Ming case, the BitTorrent (BT) file containing the copyrighted materials was published in a public online forum [5]. The second reason is that online public forums provide a convenient platform for users to exchange information, opinions and resources [4].

Online public forums, such as Hong Kong Discuss (`www.discuss.com.hk`) and Uwants (`www.uwants.com`), are Internet discussion forums that allow users to have public online conversations by posting messages. According to 2010 statistics [1], Hong Kong Discuss and Uwants were the sixth and ninth most frequently visited web sites in Hong Kong. Table 2 shows the usage statistics of these two forums.

Table 2. Usage statistics for HK Discuss and Uwants.

	HK Discuss	**Uwants**
Registered users as of February 2011	2,018,728	2,311,809
Posts as of February 2011	–	55,823,984
Online users at a particular point in time	82,513	67,950
Maximum number of online users	145,753	141,240
Posts on a weekday	–	71,642
Posts on a weekend	–	113,677

3.1 Data Collection Methodology

We created a 72-question questionnaire for data collection. The questions were divided into three sections corresponding to the principal variables: (i) online characteristics ($\mathcal{L}$); (ii) file sharing characteristics ($\mathcal{F}$); and (iii) personal particulars ($\mathcal{PP}$), which includes personalities ($\mathcal{P}$), motives ($\mathcal{M}$) and background ($\mathcal{B}$). The category facts ($\mathcal{T}$), which helps infer behavioral characteristics ($\mathcal{L}$ and $\mathcal{F}$), is omitted because in a self-reporting survey, it is deemed to be more appropriate to ask for the behavioral characteristics of an individual. However, self-reporting surveys have certain limitations, which are discussed later.

The questionnaires were posted online and were completed in an anonymous manner. It was important to ensure anonymity because sensitive information was solicited about a subject's uploading and downloading of copyrighted materials. The assurance of strict anonymity and confidentiality encouraged subjects to provide honest information, which contributed to a more accurate survey.

Online public forums were used to solicit participation in the survey. Messages were posted on three public forums: Hong Kong Discuss, Uwants and FDZone (`fdzone.forum.org`). Only those individuals who confirmed that they were 18 years or older were allowed to participate in the survey.

Over a four-month period, the system received a total of 263 questionnaire submissions, of which 133 were complete. Eliminating the questionnaires completed by individuals who had never shared copyrighted materials on the Internet left 114 complete questionnaires for analysis.

Participant Statistics The term "case" is used to describe the collection of responses in a complete questionnaire. The following are the background statistics about the 114 cases:

- **Gender:** 14.9% (17) female; 85.1% (97) male.

- **Age:** 27.2% (31) 18 to 21; 63.2% (72) 22 to 30; 9.6% (11) over 30.
- **Residence:** 69.3% (79) live in Hong Kong; 24.6% (28) live in other Chinese-speaking areas (Macau, Taiwan, Mainland China); 6.1% (7) live in other places (United States, Malaysia, Australia).
- **Marital Status:** 80.7% (92) are single; 19.3% (22) are married or in a stable relationship.
- **Type of Sharing:** 64.0% (73) have uploaded and downloaded copyrighted materials; 36.0% (41) have downloaded but not uploaded copyrighted materials.
- **Academic Qualifications:** 2.6% (3) completed (or will complete) Form 3 or lower; 25.4% (29) completed (or will complete) Form 5 to Form 7; 71.9% (82) completed (or will complete) a college degree or higher.
- **Employment Status:** 34.2% (39) are students; 53.5% (61) have stable employment; 4.4% (5) have unstable employment; 7.9% (9) are unemployed.

It is worth noting that the participants were not selected through random sampling. The statistics drawn from the data pool are not used to estimate the population parameters. In other words, statistical inference is not used to draw a conclusion such as the percentage of the population of Internet pirates that are male. Instead, the goal is to identify correlations among different sets of characteristics of an individual.

3.2 Data Preparation

Internet piracy is the copying and distribution of copyrighted files using the Internet. This definition leads to three types of Internet pirates:

- **Uploader:** Not involved in downloading; only involved in uploading (or distributing) copyrighted files
- **Downloader:** Not involved in uploading; only involved in downloading (or copying) copyrighted files
- **Sharer:** Involved in uploading and downloading copyrighted files

According to the survey results, 64% (73) of the participants are sharers, 36% (41) are downloaders, and none are purely uploaders.

The nature of Internet piracy acts leads us to divide the cases into two groups: Group I (Downloaders) and Group II (Sharers). Another

Table 3. Online characteristics.

Label	Characteristic
L_2	Online for over four hours per day on weekends
L_4	Usually visit Facebook or use MI when online
L_5	Usually go to forums when online
L_6	Visit two or more online forums
L_7	Use the same user name in different forums
L_8	Online whenever possible
L_9	Online outside working hours
L_{10}	Online during early hours
L_{11}	Online during work hours

reason for dividing the cases is that some of the variables, such as the types of the files uploaded, are not relevant to downloaders. If all the cases were to be analyzed together, the results would be more complex and more difficult to interpret.

The final questionnaires comprised 63 variables for Group I and 58 variables for Group II. Tables 3 through 5 define the variables.

3.3 Methodology

Multidimensional scaling (MDS) is a statistical technique for identifying underlying patterns or structures in a set of objects. It is useful for measuring the similarities and dissimilarities of objects and displaying the results in the form of geometric representations. One of the major uses of MDS [2] is to reveal the hidden psychological dimensions in data. In the area of criminal profiling, MDS has been used to analyze criminal behavior patterns and offender characteristics [9, 12].

MDS essentially arranges a set of objects (survey variables) on a map where the distance between two objects represents the observed correlation of the corresponding variables [2]. If an object A is in close proximity to an object B but far away from an object C, this suggests that object A and object B hold a strong relationship while a weak relationship or no relationship exists with the remote object C. Interested readers are referred to [2] for details about the mathematical and computational aspects of the MDS algorithm.

This study uses the SPSS statistical tool [11] for MDS analysis. The conceptual framework separates the variables into three sets: (i) online characteristics $\mathcal{L}$ (Table 3); (ii) file sharing characteristics $\mathcal{F}$ (Table 4); and (iii) personal particulars $\mathcal{PP}$ (Table 5).

We focus first on the behavioral characteristics. The sets $\mathcal{L}$ and $\mathcal{F}$ were analyzed using MDS. From the two-dimensional results, clusters of variables in close proximity were identified as the observed patterns of

Table 4. File sharing characteristics.

Label	Characteristic
F_2	Mostly upload songs
F_3	Mostly upload movies
F_4	Mostly upload television dramas
F_5	Mostly upload anime
F_7	Mostly upload software or games
F_8	Mostly upload pornography
F_9	Mostly download songs
F_{10}	Mostly download movies
F_{11}	Mostly download television dramas
F_{12}	Mostly download anime
F_{13}	Mostly download e-books
F_{14}	Mostly download software or games
F_{15}	Mostly download pornography
F_{16}	Use two or more file sharing systems
F_{17}	Use four or more file sharing systems
F_{18}	Buy CDs/DVDs and convert them to other digital formats for sharing
F_{19}	Download files from other sources and make changes before uploading them
F_{21}	Try to do something technical to avoid being caught
$G2 : F_{22}$	Try to do something to avoid being caught
$G1 : F_{22}$	Try to do something simple to avoid being caught
F_{23}	Never did anything to avoid being caught
F_{24}	Post or get files from Hong Kong forums or websites
F_{25}	Post or get files from Mainland Chinese forums or websites
F_{26}	Post or get files from foreign forums or websites
F_{27}	Short seeding period
F_{28}	Arbitrary seeding period
F_{29}	Mostly upload movies
F_{30}	Started using BT as a downloader for three years or more
F_{34}	Download using BT regularly
F_{35}	Download using BT irregularly
F_{39}	Have a regular uploading time when using BT
F_{41}	Use uploading tools to upload to a few locations at a time
F_{42}	Use a single file hosting service every time
F_{43}	Use different file hosting services
F_{44}	Pay for file hosting services
F_{45}	Only share files with a small group of people
F_{46}	Upload many files to file hosting services
F_{47}	Download many files from file hosting services
F_{48}	Upload files to file hosting services irregularly
F_{50}	Download files from file hosting services irregularly
F_{51}	Download files from file hosting services regularly
F_{52}	Pick uploading time arbitrarily
F_{53}	Have a regular uploading time using file hosting services
F_{54}	Download many files using eDonkey
F_{55}	Download many files using Foxy

Table 5. Personal particulars.

Label	Particular
PP_1	Live in Hong Kong
PP_2	Live in another Chinese speaking region
PP_4	Female
PP_5	Male
PP_6	Age 18 to 21
PP_7	Age 22 to 30
PP_8	Age over 30
PP_{10}	Married or in a relationship
PP_{12}	Highest academic qualification is Form 5 to Form 7
PP_{13}	Highest academic qualification is a college degree or higher
PP_{14}	Good at school
PP_{15}	Not good at school
PP_{16}	Enjoy studying
PP_{17}	Do not enjoy studying
PP_{18}	Student
PP_{19}	Stable employment
PP_{21}	Unemployed
PP_{23}	Used computers before age 16
PP_{24}	Confident about computer knowledge
PP_{25}	Employed in the computer sector
PP_{26}	Positive personality
PP_{27}	Introverted or silent personality
PP_{28}	Outgoing personality
PP_{29}	Easy-going personality
PP_{30}	Live alone
PP_{31}	Have a white-collar job
PP_{32}	Have working parent(s)
PP_{33}	Have a happy family
$G2 : PP_{34}$	Try to hide file sharing activities from family and friends
$G1 : PP_{34}$	Try to hide file sharing activities from family
$G1 : PP_{35}$	Try to hide file sharing activities from friends
PP_{36}	Reason: Want to make contributions to others
PP_{37}	Reason: Want peer recognition (earn points in discussion forums)
PP_{40}	Intend to hide identity and avoid being traced

behavioral characteristics. To examine how these behavioral characteristics are related to an individual's personal particulars, an MDS analysis was performed again on the variables of each cluster with the variables in $\mathcal{PP}$ for each group.

3.4 Results

This section describes the results of MDS profiling. The profiling revealed six types of downloaders and six types of sharers.

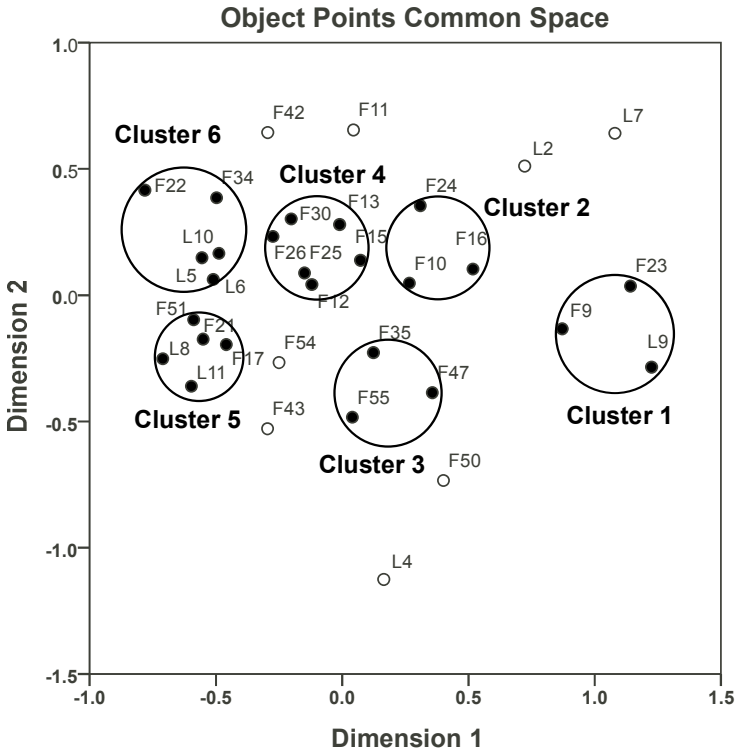


Figure 2. Group I (Downloaders): MDS of L_i and F_i.

Group I (Downloaders). For Group I, nine variables are related to online characteristics ($\mathcal{L}$), 24 variables are related to file sharing characteristics ($\mathcal{F}$), and 30 variables are related to personal particulars ($\mathcal{PP}$). Figure 2 shows the two-dimensional results of MDS analysis on the sets $\mathcal{L}$ and $\mathcal{F}$. K-means cluster analysis of the variables identified six clusters that grouped betweeen three to six characteristics.

Running an MDS analysis on each cluster with set $\mathcal{PP}$ individually yielded six two-dimensional representations. The identified behavioral characteristics and the associated personal particulars in each cluster led to the identification of the following six types of downloaders:

- **Type 1 (Music Downloaders):** These individuals are online outside of work hours (L_9). They mostly download songs (F_9) and do not do anything to avoid being caught (F_{23}).

  Type 1 downloaders are usually male (PP_5) aged 22 to 30 (PP_7). They live in Hong Kong (PP_1) and have obtained (or will obtain) a college degree or higher (PP_{13}). They have used computers since

they were young (before age 16) (PP_{23}). They usually have stable jobs (PP_{19}) and happy families (PP_{33}).

- **Type 2 (Movie Downloaders):** These individuals use two or more file sharing systems (F_{16}) to download movies (F_{10}), and post or get files from local forums or web sites (F_{24}).

  Type 2 downloaders do not have a specific set of personal particulars.

- **Type 3 (P2P Downloaders):** These individuals do not use BT regularly (F_{35}). Instead, they download files from file hosting services (F_{47}) and by using Foxy (F_{55}).

  Type 3 downloaders do not have a specific set of personal particulars.

- **Type 4 (Anime/e-Book/Porn Downloaders):** These individuals mainly download anime (F_{12}), comics, e-books (F_{13}) and pornography(F_{15}). They usually post or get files from the Chinese mainland (F_{25}) and foreign (F_{26}) forums or web sites. They are experienced BT downloaders, having used it for three years or more (F_{30}).

  Type 4 downloaders do not have a specific set of personal particulars.

- **Type 5 (Cyberlocker Downloaders):** These individuals download files from file hosting services (F_{51}) on a regular basis (e.g., every day or every weekend). They try to go online whenever possible (L_8), even during working hours (L_{11}). They use four or more file sharing systems (F_{17}) and use advanced techniques (e.g., onion routers or proxy servers) to avoid getting caught (F_{21}).

  Type 5 downloaders have outgoing personalities (PP_{28}). They usually live alone (PP_{30}) in a Chinese-speaking region outside Hong Kong (PP_2). Typically, they are over 30 (PP_8) and are unemployed (PP_{21}). They are cautious about their online identities and actively avoid being traced by others (PP_{40}). They tend to conceal their file sharing activities from family (PP_{34}) and friends (PP_{35}).

- **Type 6 (Forum Downloaders):** These individuals usually visit forums when they are online (L_5) and visit more than two online forums (L_6). They usually go online during the early hours (L_{10}). They use BT to download regularly (F_{22}) and use simple methods (e.g., disconnecting from the swarm right after a download is complete) to avoid getting caught (F_{34}).

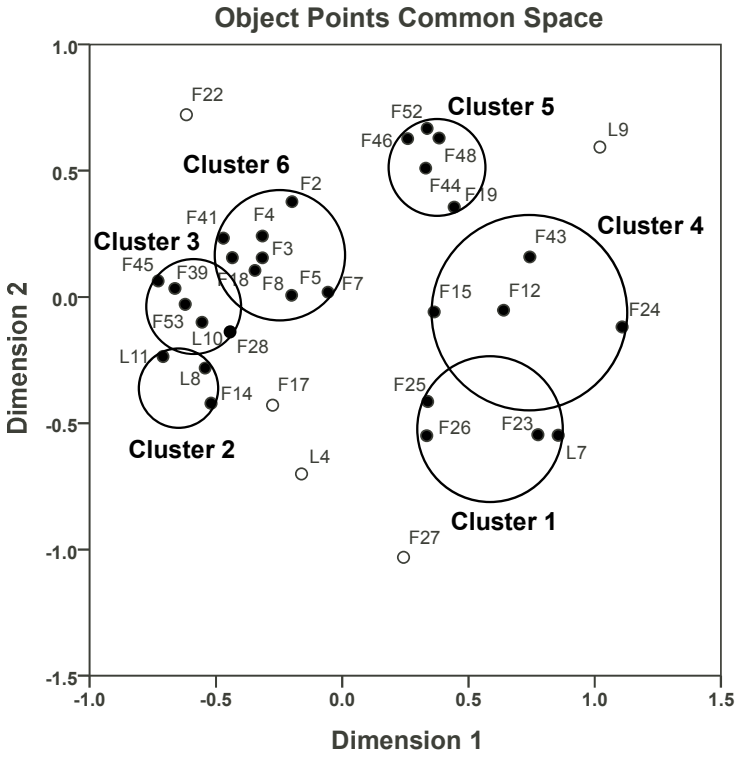


Figure 3. Group II (Sharers): MDS of L_i and F_i.

Type 6 downloaders do not have a specific set of personal particulars.

Group II: Sharers. For Group II, six variables are related to online characteristics ($\mathcal{L}$), 28 variables are related to file sharing characteristics ($\mathcal{F}$), and 24 variables are related to personal particulars ($\mathcal{PP}$).

Figure 3 shows the two-dimensional MDS analysis results for the sets $\mathcal{L}$ and $\mathcal{F}$. K-means cluster analysis identified six clusters that grouped between three and eight characteristics.

Six two-dimensional representations were obtained by running an MDS analysis on each cluster with the set $\mathcal{PP}$. The six clusters led to the identification of the following six types of sharers:

- **Type 1 (Forum Sharers):** These individuals use the same user name in different online forums (L_7) and do not take any measures to avoid getting caught (F_{23}). They usually post files to or get files from Mainland Chinese (F_{25}) and foreign (F_{26}) forums or web sites.

  Type 1 sharers do not have a specific set of personal particulars.

- **Type 2 (P2P (Public) Sharers):** These individuals go online whenever possible (L_8), even during working hours (L_{11}). They mostly download software or games (F_{14}).

  Type 2 sharers usually live alone (PP_{30}). They do not do well in school (PP_{15}) and do not enjoy studying (PP_{17}). They share files because they seek peer recognition (PP_{37}) (e.g., earn points in discussion forums).

- **Type 3 (P2P (Private) Sharers):** These individuals go online during the early hours (L_{10}). They usually share their files with a small group of people (F_{45}). When they upload files using BT, they usually stay in the swarm for arbitrary periods of time (F_{28}). They have regular uploading times when using BT (F_{39}) and file hosting services (F_{53}).

  Type 3 sharers have easy-going personalities (PP_{29}). They do not do well in school (PP_{15}) and do not enjoy studying (PP_{17}). They usually live alone (PP_{30}) and share files to gain peer recognition (PP_{37}).

- **Type 4 (Anime/e-Book/Porn Sharers):** These individuals upload anime (F_{12}) and pornography (F_{15}). They usually post files to or get files from local forums or web sites (F_{24}). They use multiple file hosting services for sharing (F_{43}).

  Type 4 sharers usually live in Hong Kong (PP_9). They do not have any other specific personal particulars.

- **Type 5 (Cyberlocker Sharers):** These individuals download files from other sources and make some changes to the files before uploading them (F_{19}). They do not regularly (F_{48}) upload many files to file hosting services (F_{46}) and are willing to pay for better transfer speeds (F_{44}). They usually do not have regular uploading times (F_{52}).

  Type 5 sharers do not have a specific set of personal particulars.

- **Type 6 (Media-Shifter Sharers):** These individuals upload most file types (songs (F_2), movies (F_3), TV dramas (F_4), anime (F_5), software and games (F_7), and pornography (F_8)). They usually buy CDs/DVDs and convert them to other digital formats for file sharing (F_{18}). They use uploading tools to upload to a few locations at the same time (F_{41}).

  Type 6 sharers have easy-going personalities (PP_{29}). They usually live alone (PP_{30}) and try to hide their file sharing activities from

family and friends (PP_{34}). They share files with others because they seek peer recognition (PP_{37}) and want to make contributions to others (PP_{36}).

4. Discussion

This section discusses the limitations of the survey and the accompanying analysis, along with plans for future research to address the limitations.

4.1 Limitations

The results of this study suggest a tendency or higher probability that relationships exist between behavioral characteristics and personal particulars. The taxonomy of Internet pirates described in this paper is neither exclusive nor exhaustive. It was derived from the responses of 114 survey participant and, as such, has the following limitations:

- **Self-Reported Data:** The online survey obtained self-reported data from volunteers. There is the possibility that the reported data may be biased. Also, it is possible that some survey participants were dishonest or were incapable of providing accurate answers to certain questions.

- **Participant Pool:** Only individuals aged 18 or above were allowed to participate in the survey. We anticipate that a large proportion of Internet pirates are younger than 18 (e.g., students from secondary schools and even primary schools). Clearly, the taxonomy would not apply to these individuals. Another issue is that the survey participants were clearly limited to Internet pirates who were interested in helping the study.

- **Cultural Dependency:** In general, the results of criminal profiling studies are highly culture-dependent. This is due to the fact that the environment surrounding an individual's life influences the individual's perceptions of issues. Therefore, the results of this study may not be applicable to non-Chinese cultures.

4.2 Future Directions

Despite the limitations of the study, the findings indicate that there are two typologies, downloaders and sharers. Future research in this area should perform empirical tests based on these typologies and the conceptual framework for profiling Internet pirates. For example, using

the typologies to predict actual Internet pirating behaviors could help determine the accuracy of the typologies.

Additionally, the applicability of the conceptual framework to other crimes, such as Internet auction fraud, should be examined.

From the design perspective, future studies should include participants from other online communities (e.g., social networks), convicted cyber criminals and even non-criminals as a control group. The use of larger and diverse populations would significantly enhance the resulting data analysis and provide better profiles of Internet criminals. Also, MDS analyses and K-means clustering comparisons could help provide quantitative evaluations of the taxonomies.

5. Conclusions

The conceptual framework for profiling Internet pirates presented in this paper incorporates six fundamental variables: observable online traces, online characteristics, file sharing characteristics, personality, motive and background. These variables are further divided into three conceptual categories: facts, behaviors and personal particulars. In the conceptual framework, behaviors can be influenced by personal particulars and inferred by observed facts. Similarly, an individual's behaviors reflect his or her personal particulars.

MDS and clustering analyses of the results of the survey of 114 Internet pirates yielded a taxonomy comprising six types of downloaders and six types of sharers, each with different sets of behavioral characteristics. Some of the downloader and sharer types are also characterized by their personal particulars.

Criminal profiling techniques have been shown to be immensely useful in traditional criminal investigations. Profiling frameworks and taxonomies for specific types of cyber crimes would aid forensic scientists and practitioners in devising effective and efficient investigation plans. As Internet piracy and other cyber crimes become even more rampant, it is crucial that law enforcement agents and other officials who combat these crimes have better understanding of the underlying criminal behaviors.

References

[1] Alexa Internet, Top sites by countries: Hong Kong, San Francisco, California (`www.alexa.com/topsites/countries/HK`), 2012.

[2] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer, New York, 2005.

[3] A. Caspi, B. Roberts and R. Shiner, Personality development: Stability and change, *Annual Review of Psychology*, vol. 56, pp. 453–484, 2005.

[4] K. Chow, K. Cheng, L. Man, P. Lai, L. Hui, C. Chong, K. Pun, W. Tsang, H. Chan and S. Yiu, BTM: An automated rule-based BT monitoring system for piracy detection, *Proceedings of the Second International Conference on Internet Monitoring and Protection*, 2007.

[5] Court of Final Appeal of the Hong Kong Special Administrative Region, Final Appeal No. 3 of 2007 (Criminal), between Chan Nai Ming and HKSAR, Hong Kong, China (`legalref.judiciary.gov.hk/lrs/common/ju/ju_frame.jsp?DIS=57111`), 2007.

[6] Envisional, An Estimate of Infringing Use of the Internet, Technical Report, Cambridge, United Kingdom (`documents.envisional.com/docs/Envisional-Internet_Usage-Jan2011.pdf`), 2011.

[7] N. Fisk, *Understanding Online Piracy: The Truth About Illegal File Sharing*, ABC-CLIO, Santa Barbara, California, 2009.

[8] J. Gantz and J. Rochester, *Pirates of the Digital Millennium: How the Intellectual Property Wars Damage Our Personal Freedoms, Our Jobs and the World Economy*, FT Prentice Hall, Upper Saddle River, New Jersey, 2005.

[9] E. Hickey, *Serial Murderers and Their Victims*, Wadsworth, Belmont, California, 2006.

[10] R. Kocsis, *Criminal Profiling: Principles and Practice*, Humana Press, Totowa, New Jersey, 2006.

[11] R. Levesque, *SPSS Programming and Data Management: A Guide for SPSS and SAS Users*, SPSS, Chicago, Illinois, 2007.

[12] G. Palermo and R. Kocsis, Offender profiling: An introduction to the sociopsychological analysis of violent crime, *Journal of the American Academy of Psychiatry and the Law*, vol. 33(3), pp. 421–423, 2005.

[13] M. Rogers, The role of criminal profiling in the computer forensics process, *Computers and Security*, vol. 22(4), pp. 292–298, 2003.

[14] B. Turvey (Ed.), *Criminal Profiling: An Introduction to Behavioral Evidence Analysis*, Academic Press, Oxford, United Kingdom, 2012.

Chapter 19

REAL-TIME COVERT TIMING CHANNEL DETECTION IN NETWORKED VIRTUAL ENVIRONMENTS

Anyi Liu, Jim Chen and Harry Wechsler

Abstract Despite extensive research on malware and Trojan horses, covert channels are still among the top computer security threats. These attacks, which are launched using specially-crafted content or by manipulating timing characteristics, transmit sensitive information to adversaries while remaining undetected. Current detection approaches typically analyze deviations from legitimate network traffic statistics. These approaches, however, are not applicable to highly dynamic, noisy environments, such as cloud computing environments, because they rely heavily on historical traffic and tedious model training. To address these challenges, we present a real-time, wavelet-based approach for detecting covert timing channels. The novelty of the approach comes from leveraging a secure virtual machine to mimic a vulnerable virtual machine. A key advantage is that the detection approach does not require historical traffic data. Experimental results demonstrate that the approach exhibits good overall performance, including a high detection rate and a low false positive rate.

Keywords: Covert timing channels, detection, virtual environments

1. Introduction

Covert timing channels are among the most prevalent computer security attacks. These attacks exfiltrate sensitive information and credentials, such as secret keys and passwords, by manipulating the timing or the ordering of network events [6]. A successful covert timing channel leaks sensitive information to external attackers without triggering alerts even in well-protected networks.

Most covert timing channel detection approaches use signatures to detect known channels [5] or consider anomalous deviations from legit-

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 273–288, 2013.

imate network traffic to detect unknown channels [3, 5, 6, 17]. These approaches, however, have at least two limitations. First, their effectiveness depends on the availability of a sufficient amount of legitimate (or attack-free) traffic to construct attack signatures and accurate models of legitimate traffic. Unfortunately, in a networked virtual environment, where virtual machines are interconnected, legitimate traffic is either hard to obtain or contains noise due to the imprecise timekeeping mechanism used in virtual machines [19]. This greatly reduces the effectiveness of signature-based and anomaly-based approaches for covert timing channel detection.

The second limitation is that most existing approaches were designed to detect covert timing channels that transmit information at a high rate to achieve high bandwidth. However, an attacker can deliberately extend the duration of the covert transmission process. This renders the timing patterns of covert timing channel traffic almost indistinguishable from those of legitimate network traffic. Thus, existing approaches are generally unable to detect slow covert timing channels (e.g., the channels described in [7, 13]).

This paper presents a new wavelet-based approach for detecting covert timing channels in networked virtual environments, where no historical data pertaining to legitimate traffic is available. The detection approach employs distance metrics between outbound traffic generated by two virtual machines, a suspicious virtual machine and a benign virtual machine. More specifically, the metrics use a discrete wavelet-based multiresolution transformation (DWMT) to measure the variability of timing differences at all decomposition scales. To our knowledge, this is the first online detection approach that operates by quantitatively measuring the distance between two network flows.

The wavelet-based approach is evaluated using a series of experiments focused on detecting several covert timing channels, including their slow variations. The robustness of the detection approach in the presence of noise is also investigated. Experimental results demonstrate that the wavelet-based approach is very effective at detecting covert timing channels in real time, despite efforts to make them slow and stealthy.

2. Background

Covert timing channels have been the subject of much research. Berk, *et al.* [3] implemented a simple binary covert timing channel based on the Arimoto-Blahut algorithm [4], which computes the input distribution that maximizes channel capacity. Cabuk [5] was the first to describe an IP covert timing channel (IPCTC) and a more advanced traffic replay

covert timing channel (TRCTC). Shah, *et al.* [17] developed JitterBug, a keyboard device that slowly leaks information typed by a user over the Internet. Giffin, *et al.* [8] showed that low-order bits of TCP timestamps can be exploited to create a covert timing channel due to the shared statistical properties of timestamps and packet timing.

Covert timing channels can also be used to trace suspicious traffic. For example, Wang, *et al.* [20] leveraged well-designed inter-packet delays to trace VoIP traffic. Their recent work [16] utilizes watermarked network traffic to trace covert timing channels back to botmasters. Gianvecchio, *et al.* [7] and Liu, *et al.* [13] designed model-based covert channel encoding schemes that seek to achieve both undetectability and robustness.

A number of covert timing channel detection methods have been developed. Peng, *et al.* [14] showed that the Kolmogorov-Smirnov test is effective at detecting covert timing channels that manipulate inter-packet delays. Cabuk [5] investigated a regularity-based approach for detecting covert timing channels. In particular, he proposed an ϵ-similarity metric to measure the proportion of similar inter-packet delays. The limitation of the ϵ-similarity metric is that it only targets IPCTCs; it is not general enough to detect other covert timing channels. Berk, *et al.* [3] employed a simple mean-max ratio test to detect binary and multi-symbol covert timing channels. However, the mean-max ratio test assumes that legitimate inter-packet delays follow a normal distribution, which is often not true for real-world traffic.

Gianvecchio and Wang [6] proposed an effective entropy-based method for detecting covert timing channels. However, our approach has three advantages over this method. First, as an essential step for computing patterns of length m, the corrected conditional entropy metric used by Gianvecchio and Wang has quadratic time complexity, while our approach has linear time complexity. Second, the corrected conditional entropy metric cannot detect stealthy covert timing channels, while our wavelet-based distance metric can detect a variety of covert timing channels, including their stealthy versions. Third, our approach can be used in an online configuration and is well suited to virtual environments.

Jing and Wang [10] developed a wavelet-based method for measuring time distortion in low latency anonymous networks. This method is closely related to our approach. However, our approach has two advantages. First, unlike the time distortion metric, our wavelet-based distance metric can clearly differentiate between slow versions of covert timing channels. Second, our approach is specifically designed to detect covert timing channels, which have more stealthy timing characteris-

tics than the timing distortion of packet transformation introduced by anonymous networks.

Zhang *et al.* [22] have proposed online covert timing channel prevention mechanisms that mitigate information leakage. Although their mechanisms are efficient at identifying the upper and lower bounds of information leakage as a function of elapsed time, they are unable to detect covert timing channels.

3. Wavelet-Based Detection Approach

This section describes a wavelet-based metric for quantitatively measuring the distance between the inter-packet delays (IPDs) of a legitimate flow and a covert timing channel flow. The section begins by describing the covert timing channel detection approach. Next, it discusses how the timing distance between two outbound network flows can be measured using the variables derived from a wavelet-based multi-resolution transformation (DWMT). Finally, it formulates the metric for measuring the timing distance between network flows.

Given an inbound network flow I, the problem of measuring the timing distance between two outbound flows, O_1 and O_2, can be formulated as follows. The inbound flow I contains $K > 0$ packets $\langle p_{i,1}, \ldots, p_{i,K-1} \rangle$. In response to I, a virtual machine VM1 generates O_1 containing $M > 0$ packets $\langle p_{o_1,1}, \ldots, p_{o_1,M-1} \rangle$ and a virtual machine VM2 generates O_2 containing $N > 0$ packets $\langle p_{o_2,1}, \ldots, p_{o_2,N-1} \rangle$. Since the packets in O_i were generated by a virtual machine in response to I, the packets in I and O_i are segmented based on their request/response relationship. Specifically, for the j^{th} inbound segment $I^j = \langle p^j_{i,1}, \ldots, p^j_{i,m} \rangle$, the responding outbound segment in O_i is defined as $O^j_i = \langle p^j_{o_i,1}, \ldots, p^j_{o_i,o} \rangle$. The notation $t^j_{(o_i,k)}$ is used to represent the timestamp of the k^{th} packet in the j^{th} segment of O_i. Since the length of I is much greater than that of O_i, O_i can be further aggregated into w aggregated segments.

3.1 Covert Timing Channel Detection

Figure 1 shows example inbound and outbound flows. O^{i-2}_2 is the responding outbound segment of I^{i-2} in flow O_2. For $O_{1,i}$, the packets in segment t_i are denoted as $\langle p_{o_1,0}, \ldots, p_{o_1,m-1} \rangle (m > 0)$. Similarly, for $O_{2,i}$, the packets in the segment are denoted as $\langle p_{o_2,0}, \ldots, p_{o_2,n-1} \rangle (n > 0)$, where $n \approx m$. The timestamps of the j^{th} packet in $O_{1,i}$ and $O_{2,i}$ are denoted as $t^j_{(O_1,i)}$ and $t^j_{(O_2,i)}$, respectively.

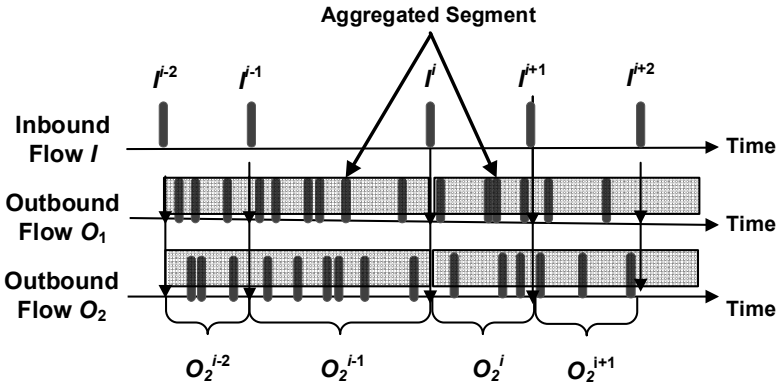


Figure 1. Inbound and outbound flows.

3.2 Timing Distance

The assumption underlying the timing distance metric is that the timing patterns of outbound legitimate traffic are similar. Therefore, by characterizing the timing patterns of network flows, it is possible to efficiently measure the timing distance between a legitimate flow and a covert timing channel flow. The measurement is performed using a discrete wavelet-based multi-resolution transformation (DWMT) [1]. The DWMT, which has been widely used in signal processing and anomaly detection [10], has at least three prominent features. First, DWMT provides multi-resolution analysis, which facilitates the examination of a sequence of data at different scales. Second, DWMT allows feature localization in that it provides information about the characteristics of a signal and "approximately" where they occur in time. Third, DWMT supports online analysis, i.e., online comparisons of the differences between two flows.

The DWMT takes a sequence of data as input and transforms the sequence into a number of wavelet coefficient sequences. Specifically, the l-level DWMT takes a sequence of IPDs and transforms the sequence into: (i) l-wavelet detailed coefficient vectors at different scales (CD_i, where $1 \leq i \leq l$); and (ii) a low-resolution approximate vector (CA_l, where $1 \leq i \leq l$). For the j^{th} segment of O_i, the wavelet detailed coefficients vector at scale l can represented as:

$$V(i,j,l) = \langle CD_l^j(o_i,1), ..., CD_l^j(o_i,N_j)\rangle$$

where $N_j = n_j \times 2^{-j}$ is the number of wavelet coefficients at scale j, and $c_{l,k} = c_{l-1,2k} + c_{l-1,2k+1}$ for $l \geq 0$.

Our wavelet-based distance (WBD) must satisfy three design goals. First, the WBD between two legitimate flows should be small. Second, the WBD between a legitimate flow and a covert timing channel flow should be different enough to be detectable. Third, the WBD should be able to differentiate between a regular covert timing channel and a stealthy covert timing channel.

To achieve these goals, we define three derived vectors based on the coefficient vector $V(i,j,l)$ at scale l ($l \geq 0$):(i) intra-flow vector ($intraFV$); (ii) inter-flow vector ($interFV$); and (iii) Kullback-Leibler divergence (KLD) vector.

We define:

$$\begin{aligned} intra(i,j,l) = \langle CD_l^j(O_i,1) &- CD_l^j(O_i,0),...,CD_l^j(O_i,N_j) \\ &- CD_l^j(O_i,N_{j-1})\rangle \end{aligned}$$

which reflects the fluctuating characteristics between adjacent coefficients in a coefficient vector at scale j of one flow O_i. $intraFV(j,l)$ is defined as the Euclidean distance between $intra(1,j,l)$ and $intra(2,j,l)$:

$$intraFV(j,l) = dist(intra(1,j,l), intra(2,j,l)).$$

Similarly, we define:

$$interFV(j,l) = dist(V(1,j,l), V(2,j,l))$$

as the Euclidean distance between the two coefficient vectors $V(1,j,l)$ and $V(2,j,l)$, which characterizes the deviation between two wavelet coefficients for the same segment j at the same scale j.

The Kullback-Leibler divergence (KLD) has been used to measure the distance between two probability distributions $p_1(x)$ and $p_2(x)$ [11]. From an information theory perspective, KLD measures the expected number of extra bits required to code samples from $p_1(x)$ when using a code based on $p_2(x)$. The KLD for the probability distributions $p_1(x)$ and $p_2(x)$ is given by:

$$KLD(p_1(x), p_2(x)) = \sum_{i=0}^{|x|} p_1(x) \log \frac{p_1(x)}{p_2(x)}. \quad (1)$$

In order to calculate the KLD between two wavelet coefficient vectors at scale j, it is necessary to obtain the probability distribution of $V(i,j,l)$, i.e., $p(V(1,j,l))$. To obtain $p(V(1,j,l))$, the term $V(i,j,l) = \langle CD_l^j(o_i,1),\ldots,CD_l^j(o_i,N_j)\rangle$ with numeric coefficients is converted to a vector of symbols $\tilde{S}_i = \langle \alpha_1,\ldots,\alpha_l\rangle$. Then, $p(V(i,j,l))$ is calculated

based on $\tilde{S}_i$. The effectiveness of converting $V(i,j,l)$ to $\tilde{S}_i$ depends on a function F that maps $CD_l^j(o_i,m)$ to an alphabet $\mathcal{A} = \alpha_1, \ldots, \alpha_m$, where the soundness of translation from $CD_l^j(o_i,m)$ to α_m is given by:

$$F(CD_l^j(o_i,m)) = \alpha_m$$

if and only if $\beta_{j-1} \leq \alpha_m \leq \beta_j (1 \leq i \leq l; 1 \leq j \leq m)$.

To facilitate effective conversion, the data discretization scheme of SAX [12] is used to assign wavelet coefficients to k equiprobable regions. Each continuous coefficient value that falls in a region maps to a unique symbol α_i $(1 \leq i \leq \kappa)$. The KLD defined in Equation (1) for $V(1,j,l)$ and $V(2,j,l)$ now becomes:

$$KLD(j,l) = KLD(p(V(1,j,l)), p(V(2,j,l))) = \sum_{i=0}^{k} p(\tilde{S}_1) \log \frac{p(\tilde{S}_1)}{p(\tilde{S}_2)}.$$

A tradeoff is involved when choosing the optimal size κ of alphabet $\mathcal{A}$. A large value κ keeps more information about the distribution of continuous data, but it may generate too many false alarms. In contrast, a small value of κ keeps less information about the distribution, but it may be difficult to detect slow and stealthy attacks. We choose $\kappa = 10$ to retain the ability to measure the deviation of malicious traffic. All the WBD values calculated in this paper use $\kappa = 10$, except when stated otherwise.

Given $intraFV$, $interFV$ and KDL at scale l, the wavelet-based distance (WBD) between O_1 and O_2 for all scales is defined by:

$$WBD(O_1,O_2) = \sum_{j=1}^{m} \left(\sum_{l=0}^{m} intraFV(j,l) \times \sum_{l=0}^{m} interFV(j,l) \times \sum_{l=0}^{m} KLD(j,l) \right)^2.$$

WBD essentially summarizes the divergence of intra-flow, inter-flow and KLD between two network flows. A large value of WBD indicates a significant difference between two flows while a small value of WBD implies that the two flows are similar.

4. Implementation

Figure 2 shows the architecture of the detection system. It has three main components: (i) a traffic filter, which identifies the packets that are

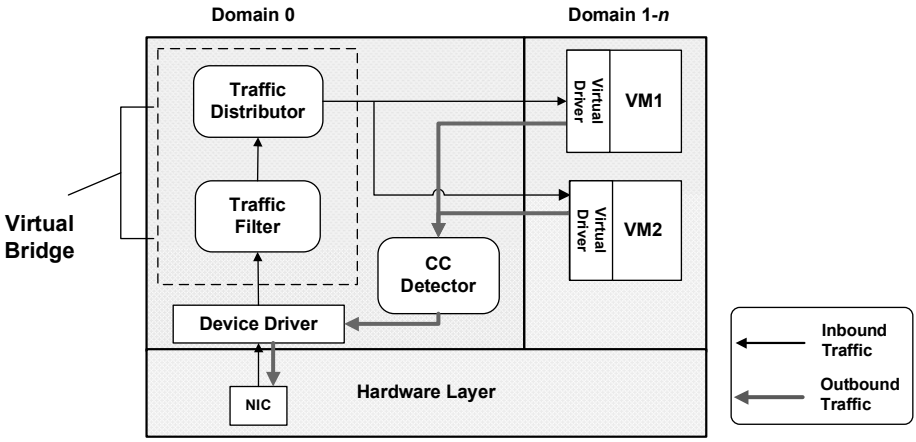


Figure 2. Detection system architecture.

sent to the suspicious virtual machine; (ii) a traffic distributor, which forwards inbound packets to the suspicious virtual machine and the benign virtual machine; and (iii) a covert channel (CC) detector, which uses the timing distance metric to calculate the timing distance between the suspicious virtual machine and the benign virtual machine.

Our covert channel detection system is implemented in C on top of Xen [2]. The traffic filter and traffic distributor are implemented as a virtual bridge. To emulate a malicious program that exfiltrates insider information, we modified the source code of `vsftpd v2.3.4` running on a Real Time Application Interface (RTAI) for Linux. The modified `vsftpd` code incorporates a covert timing channel encoder that generates a timing delay before sending a packet to a client. The covert timing channel encoder is implemented in C and inline assembly code that invokes instructions to the read timestamp counter (RDTSC) of a CPU. The RDTSC instruction was chosen because it has excellent resolution and requires low overhead for keeping time information [7].

5. Evaluation

The effectiveness of our approach was evaluated using the WBD metric to detect a number of covert timing channels. The covert timing channels, which are shown in Table 1, include both active and passive channels. The following sections describe the detection methods and the detection results for each type of covert timing channel.

Table 1. Covert timing channels used in the evaluation.

Name	**Attack Description**	**Active/ Passive**	**Detectable?**
IP Channel (IPCTC) [5]	Transmit `1`-bit or `0`-bit by choosing to send or not send a packet during time interval w	Passive	Yes
Time-Replay Channel (TRCTC) [5]	Transmit `1`-bit or `0`-bit by choosing historical legitimate IPDs as additional input from different baskets	Active	Yes
Botnet Traceback Watermark (BTW) [16]	Inject modified control text	Passive	Yes
JitterBug [17]	Operate small delays in keystrokes to affect the original IPDs	Passive	Yes

5.1 Detection Methods

Four broad classes of measurements were used to detect covert timing channels: (i) statistical tests; (ii) timing distortion metric for measuring low latency anonymous networks [10]; (iii) corrected conditional entropy metric [6]; and (iv) our wavelet-based distance (WBD) metric.

Four statistical tests were employed: (i) shape tests; (ii) Kolmogorov-Smirnov test (KS-Test) [9]; (iii) Welch's t-test (WT-Test) [21]; and (iv) regularity test (RT-Test) [5].

The shape tests employ first-order statistics such as mean, standard deviation and empirical cumulative distribution.

The KS-Test measures the distance between a legitimate sample and a test sample. A small test score implies that the test sample is close to the legitimate sample while a large test score indicates the possible presence of a covert timing channel.

The WT-Test determines if the means of two samples with different sizes and variances are different. The test statistic for the WT-Test is computed as:

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{{s_1}^2}{N_1} + \frac{{s_2}^2}{N_2}}}$$

where $\overline{X_i}$, ${s_i}^2$ and N_i are the i^{th} sample mean, variance and sample size, respectively.

The RT-Test determines if the variance of inter-packet delays is relatively constant. In our evaluation, the sequence of outbound IPDs is divided into k segments, following which the standard deviation of segment i (σ_i) is computed. The regularity score is computed as:

$$regularity = STDEV(\frac{|\sigma_i - \sigma_j|}{\sigma_i}\ i < j, \forall i, j).$$

Jin and Wang [10] proposed a wavelet-based metric (MLAN) that quantitatively measures the practical effectiveness of anonymous networks in the presence of timing attacks. This metric can be used to determine the likelihood that a network flow has been distorted by a covert timing channel. MLAN is computed as:

$$e_j = \frac{\sum_{p=0}^{N_j-1}[CD(X,Y,j)(p)]^2}{N_j}$$

where $CD(X,Y,j)(p)$ is the p^{th} $(p = 0, \ldots, N_j - 1)$ wavelet detail coefficient at scale j for the j^{th} vector $D(X,Y,j)$ and $N_j = 2^{-j}n_j$ is the number of wavelet detail coefficients at scale j.

Gianvecchio and Wang [6] have shown that the corrected conditional entropy (CCE) is effective at detecting different types of covert timing channels. The CCE is computed as:

$$CCE(x_m|x_1, \ldots, x_{m-1}) = CE(x_m|x_1, \ldots, x_{m-1}) + perc(x_m)EN(x_1)$$

where $CE(x_m|x_1, \ldots, x_{m-1})$ is the estimated conditional entropy of the pattern $x_1, \ldots, x_m$; $perc(x_m)$ is the percentage of unique patterns of length m; and $EN(x_1)$ is the entropy with m fixed at one (i.e., first-order entropy only).

5.2 IPCTC Detection

The first set of experiments was designed to test the ability of our approach to detect the presence of IPCTCs [5] (see Table 1 for the mechanism). An IPCTC encodes a `1`-bit by transmitting a packet during a timing interval w, and encodes a `0`-bit by not transmitting a packet during w. In our experiments, the covert timing channel encoder read the `/etc/passwd` file and encoded its binary data into an IPCTC.

Four encoding schemes were used. Given the observation that the average IPD of traffic was 0.147s, we designed the first three encoding schemes by choosing timing intervals w of 0.1s (IPCTC1), 0.12s (IPCTC2) and 0.14s (IPCTC3). The fourth encoding scheme (IPCTC4) rotated between the three w timing intervals after every 100 packets to

Table 2. Test scores of legitimate and IPCTC IPDs.

	Legit.	**IPCTC1**	**IPCTC2**	**IPCTC3**	**IPCTC4**
Average #IPDs	0.1472	0.1560	0.1870	0.2180	0.1880
Mean	0.1472	0.1560	0.1870	0.2180	0.1880
Std Dev	0.041	0.089	0.106	0.124	0.111
Regularity	0.9723	0.9723	0.9723	0.9723	0.9723
WT-Test	0	1	1	1	1
KS-Test	0	1	1	1	1
KS-Test (p value)	0.9670	0	0	0	0
MLAN	0.0433	1.3616	1.5072	1.7034	1.353
CCE	1.1874	0.6055	0.6095	0.6099	0.7614
WBD	0.6295	15,982.68	37,092.88	76,188.46	50,241.94

avoid creating a regular pattern of inter-packet delays. This design ensured that both legitimate and covert timing channel flows have almost the same duration and send almost the same number of packets at run time.

The test was run 100 times for durations of 50 seconds. Around 400 packets were collected in each flow. For the regularity test, the sequence of inter-packet delays was divided into 20 segments. According to Gianvecchio and Wang [6], an IPCTC is the easiest covert timing channel to detect because its abnormality shows up in simple statistical tests.

Table 2 shows the detailed results for all the test flows. Although all the IPCTCs were detected in the tests, the WBD metric produced the best results. Note that the WBD of legitimate flows is very small (0.6295), which is in stark contrast to the WBDs of IPCTC flows, all of which are 15,982.68 or higher. Although MLAN and CCE are able to differentiate all four IPCTCs from legitimate flows, it is almost impossible to differentiate between the various IPCTCs. For example, the CCE values for all four IPCTCs range from 0.6055 to 0.7614, with three of them close to 0.6.

5.3 TRCTC Detection

The second set of experiments investigated the ability of our approach to detect TRCTCs [5]. Compared with an IPCTC, a TRCTC is a more advanced covert timing channel that replays a set of legitimate inter-packet delays to mimic the behavior of legitimate traffic. A TRCTC transmits a `0`-bit by randomly replaying an inter-packet delay from Bin_0 and transmits a `1`-bit by randomly replaying an inter-packet delay from Bin_1. Since Bin_i ($i = 0$ or 1) contains legitimate traffic, the distri-

Table 3. Test scores of legitimate and TRCTC IPDs.

	Legit.	**TRCTC1**	**TRCTC2**	**TRCTC3**	**TRCTC4**
Mean	0.1472	0.1546	0.1471	0.1466	0.1467
Std Dev	0.0406	0.0411	0.0409	0.0410	0.0411
Regularity (size = 20)	0.2453	0.2958	0.1991	0.3009	0.1955
WT-Test	0	0	0	0	0
KS-Test	0	0	0	0	0
KS-Test (p value)	0.9670	0.6451	0.9982	1	1
MLAN	0.0433	0.8700	1.0074	0.8995	0.8561
CCE	1.1833	1.1589	1.1749	1.1813	1.1829
WBD	0.4669	859.58	651.92	455.50	318.91

bution of TRCTC traffic is approximately equal to the distribution of legitimate traffic. The TRCTC was encoded and transmitted bit by bit to the receiver. Then, the message was rebuilt by the decoder bit by bit as in the IPCTC experiments.

Four versions of TRCTCi ($i = 1, \ldots, 4$) were created by injecting one additional bogus packet in the flow after every a packets ($a = 5$, 10, 15 and 20). Note that a larger value of a indicates a slower attack.

The TRCTC results shown in Table 3 are similar to those obtained in the IPCTC experiments. Since TRCTCs have the same distribution as legitimate traffic, the WT-Test and the KS-Test failed to detect the TRCTCs – the tests accept the null hypothesis that the IPDs of TRCTC traffic and legitimate traffic have the same distribution. Although MLAN and CCE tests can differentiate TRCTCs from legitimate traffic, they are incapable of identifying slow TRCTCs – the aggressive TRCTC (TRCTC1) and the stealthy TRCTC (TRCTC4) yield similar results for the MLAN and CCE tests. On the other hand, the WBD test detects all the TRCTCs, including the stealthy TRCTC that has the lowest score.

5.4 BTW Detection

BackTrack watermark (BTW) is a passive covert timing channel that is embedded in normal network traffic to track communications from a bot back to its botmaster [16]. We extended the covert timing channel design to generate slow attacks. In particular, we used $2a$ ($a \geq 1$) packets to encode a bit sequence S, where the parameter a is a constant or variable generated by a pseudorandom number generator. P_{r_i} and P_{e_i} were chosen from $2a$ packets. The parameter a serves as an "amplifier," specifying the speed at which data is transmitted via the covert channel. The larger the value of a, the slower the attack. Our experiments used

Table 4. Test scores of legitimate and BTW IPDs.

	Legit.	**BTW1 (a=5)**	**BTW2 (a=10)**	**BTW3 (a=20)**	**BTW4 (a=30)**
Mean	0.1474	0.2207	0.1814	0.1627	0.1587
Std Dev	0.0399	0.1775	0.1304	0.0918	0.0828
Regularity (size = 20)	0.2453	1.3504	0.7595	0.9206	0.9723
WT-Test	0	1	1	1	0
KS-Test	0	1	0	0	0
KS-Test (p value)	0.9670	0.0004	0.2808	0.9862	0.9999
MLAN	0.0433	1.8296	1.2081	1.4790	1.2394
CCE	1.1848	1.0440	1.0842	1.1079	1.1312
WBD	2.6757	1,489.90	828.76	250.86	124.97

four versions of BTW, each with a different value of a. A total of 60 seconds of live traffic was generated by the virtual machines.

The experimental results in Table 4 show that legitimate and BTW IPDs have similar means and standard deviations, especially for higher values of a. The WT-Test and KS-Test scores can be used to detect aggressive BTWs (BTW1), but they fail to detect BTW4, the stealthiest covert timing channel in the set of experiments. Note also that the MLAN test produces a large percentage of false positive errors because it is not very effective at discriminating between legitimate and BTW flows. In general, the CCE and WBD tests yield good results – they are able to detect all the BTWs while yielding a 100% true positive rate and a zero false positive rate. However, the WBD metric yields better results than CCE because, unlike CCE, it is able to quantify the stealth of BTWs – the larger the WBD score, the more aggressive the covert timing channel.

5.5 JitterBug Detection

JitterBug is a passive covert timing channel [17] that manipulates network traffic. It operates by creating small delays in key-presses that affect the inter-packet delays of an application. It transmits a `1`-bit by increasing an IPD to a value modulo w milliseconds and transmits a `0`-bit by increasing an IPD to a value modulo $\lfloor \frac{w}{2} \rfloor$ milliseconds. For small values of w, the distribution of JitterBug traffic is very similar to that of the original legitimate traffic. However, because of the small value of w, it can cause the covert timing channel to be indistinguishable from legitimate traffic containing noise.

A value of $w = 100$ milliseconds was chosen in our experiments. Four versions of JitterBug were employed, each with a different amplifier

Table 5. Test scores of legitimate and JitterBug IPDs.

	Legit.	**JB1**	**JB2**	**JB3**	**JB4**
Mean	0.1472	0.1626	0.1549	0.1510	0.1497
Std Dev	0.0406	0.0502	0.0482	0.0432	0.0432
Regularity (size = 20)	0.2452	0.2125	0.1848	0.2453	0.2969
WT-Test	0	1	0	0	0
KS-Test	0	1	0	0	0
KS-Test (p value)	1-E7.5	1-E7.5	1-E7.5	1-E7.5	1-E7.5
MLAN	0.0433	0.6380	0.6957	0.8286	0.0710
CCE	1.1834	1.2012	1.2015	1.1933	1.1944
WBD	7.5E-05	0.0218	0.0341	0.0349	0.0645

value: JitterBug1 ($a = 5$), JitterBug2 ($a = 10$), JitterBug3 ($a = 20$) and JitterBug4 ($a = 30$). This design ensured that legitimate and covert timing channel flows have almost the same duration and send about the same number of packets at run time. A total of 30,000×10 packets were collected in real time to evaluate covert timing channel detection and the true/false positive rates.

Table 5 shows the resulting test scores. The mean and standard deviation of legitimate traffic (0.1472 ± 0.0406) and stealthy covert timing traffic are very similar, especially for JitterBug4 (0.1497 ± 0.0432). The WT-Test and the KS-Test failed to detect JitterBugs, except for the most aggressive one (JitterBug1). In the case of the regularity test, the smaller scores indicate the presence of covert timing channels for the aggressive JitterBug1 and JitterBug2, but the scores are unable to discern the stealthy JitterBug3 and JitterBug4 channels. The CCE test is unable to differentiate between the different versions of Jitterbugs because most of the scores are between 1.18 and 1.20. However, the WBD test is able to detect the stealthiest JitterBug channel (JitterBug4) because its score is 0.0645, which is much larger than the score for legitimate traffic.

6. Conclusions

The wavelet-based approach described in this paper is well suited to detecting covert timing channels in real time. A key advantage is that the detection approach does not require historical traffic data. Experimental results demonstrate that the approach exhibits good performance for a variety of covert timing channels in networked virtual environments, including slow and stealthy channels. Furthermore, the detection approach is robust in the presence of the inaccurate timekeeping mechanisms used by virtual machines.

References

[1] A. Akansu and P. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands and Wavelets*, Academic Press, San Diego, California, 2001.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, Xen and the art of virtualization, *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pp. 164–177, 2003.

[3] V. Berk, A. Giani and G. Cybenko, Covert channel detection using process query systems, *Proceedings of the Second Annual Workshop on Flow Analysis*, 2005.

[4] R. Blahut, Computation of channel capacity and rate-distortion functions, *IEEE Transactions on Information Theory*, vol. 18(4), pp. 460–473, 1972.

[5] S. Cabuk, Network Covert Channels: Design, Analysis, Detection and Elimination, Ph.D. Dissertation, Department of Computer Science, Purdue University, West Lafayette, Indiana, 2006.

[6] S. Gianvecchio and H. Wang, Detecting covert timing channels: An entropy-based approach, *Proceedings of the Fourteenth ACM Conference on Computer and Communications Security*, pp. 211–230, 2007.

[7] S. Gianvecchio, H. Wang, D. Wijesekera and S. Jajodia, Model-based covert timing channels: Automated modeling and evasion, *Proceedings of the Eleventh International Symposium on Recent Advances in Intrusion Detection*, pp. 211–230, 2008.

[8] J. Giffin, R. Greenstadt, P. Litwack and R. Tibbetts, Covert messaging through TCP timestamps, *Proceedings of the Second International Conference on Privacy Enhancing Technologies*, pp. 194–208, 2002.

[9] M. Hollander and D. Wolfe, *Nonparametric Statistical Methods*, John Wiley, New York, 1999.

[10] J. Jin and X. Wang, On the effectiveness of low-latency anonymous network in the presence of timing attack, *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 429–438, 2009.

[11] S. Kullback and R. Leibler, On information and sufficiency, *Annals of Mathematical Statistics*, vol. 22(1), pp. 79–86, 1951.

[12] J. Lin, E. Keogh, L. Wei and S. Lonardi, Experiencing SAX: A novel symbolic representation of time series, *Data Mining and Knowledge Discovery*, vol. 15(2), pp. 107–144, 2007.

[13] Y. Liu, D. Ghosal, F. Armknecht, A. Sadeghi, S. Schulz and S. Katzenbeisser, Hide and seek in time: Robust covert timing channels, *Proceedings of the Fourteenth European Conference on Research in Computer Security*, pp. 120–135, 2009.

[14] P. Peng, P. Ning and D. Reeves, On the secrecy of timing-based active watermarking trace-back techniques, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 335–349, 2006.

[15] M. Pereyra and L. Ward, *Harmonic Analysis: From Fourier to Wavelets*, American Mathematical Society, Providence, Rhode Island, 2012.

[16] D. Ramsbrock, X. Wang and X. Jiang, A first step towards live botmaster traceback, *Proceedings of the Eleventh International Symposium on Recent Advances in Intrusion Detection*, pp. 59–77, 2008.

[17] G. Shah, A. Molina and M. Blaze, Keyboards and covert channels, *Proceedings of the Fifteenth USENIX Security Symposium*, 2006.

[18] United States Government Accountability Office, Cyberspace: United States Faces Challenges in Addressing Global Cybersecurity and Governance, Report to Congressional Requesters, GAO-10-606, Washington, DC, 2010.

[19] VMWare, Timekeeping in VMWare virtual machines, Palo Alto, California (`www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf`), 2011.

[20] X. Wang, S. Chen and S. Jajodia, Tracking anonymous peer-to-peer VoIP calls on the Internet, *Proceedings of the Twelfth ACM Conference on Computer and Communications Security*, pp. 81–91, 2005.

[21] B. Welch, The generalization of student's problem when several different population variances are involved, *Biometrika*, vol. 34(1-2), pp. 28–35, 1947.

[22] D. Zhang, A. Askarov and A. Myers, Predictive mitigation of timing channels in interactive systems, *Proceedings of the Eighteenth ACM Conference on Computer and Communications Security*, pp. 563–574, 2011.

VI

CLOUD FORENSICS

Chapter 20

IMPACT OF CLOUD COMPUTING ON DIGITAL FORENSIC INVESTIGATIONS

Stephen O'Shaughnessy and Anthony Keane

Abstract As cloud computing gains a firm foothold as an information technology (IT) business solution, an increasing number of enterprises are considering it as a possible migration route for their IT infrastructures and business operations. The centralization of data in the cloud has not gone unnoticed by criminal elements and, as such, data centers and cloud providers have become targets for attack. Traditional digital forensic methodologies are not well suited to cloud computing environments because of the use of remote storage and virtualization technologies. The task of imaging potential evidence is further complicated by evolving cloud environments and services such as infrastructure as a service (IaaS), software as a service (SaaS) and platform as a service (PaaS). The implementation of forensics as a service (FaaS) appears to be the only workable solution, but until standards are formulated and implemented by service providers, the only option will be to use traditional forensic tools and rely on service level agreements to facilitate the extraction of digital evidence on demand. This paper explores the effect that cloud computing has on traditional digital forensic investigations and proposes some approaches to help improve cloud forensic investigations.

Keywords: Cloud computing, cloud forensics, digital forensic investigations

1. Introduction

Cloud computing is a rapidly evolving technological solution and business model as evidenced by the upsurge in the global adoption of cloud services. While cloud computing has its origins in mainframe computing and shares similarities with traditional Internet hosting, the ways in which cloud services are offered differ considerably. Cloud consumers can avail of self-provisioning, auto scaling and pay-per-use through ser-

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 291–303, 2013.

vices that offer increased availability, performance and scalability. In this regard, cloud computing is an evolutionary step in the provisioning of services on the Internet, allowing organizations to easily outsource their information technology requirements and pay only for the services they use. Cloud service providers such as Google, Amazon and Microsoft are driving expansion by turning their excess capacity into a business pay-per-use model that offers scalable information technology resources. This expansion is also facilitated by the availability of high-speed broadband connectivity and low-cost access from service providers.

The vast supply of anonymous computing resources in the cloud potentially provides a breeding ground for computer crime. Garfinkel [6] notes that sensitive information such as credit card data and social security numbers stored in the cloud render it an attractive target for thieves. Furthermore, cloud computing resources such as easy-to-use encryption technology and anonymous communication channels reduce the likelihood that the nefarious activities undertaken by criminal elements are intelligible to law enforcement.

The cloud can also be used as an instrument to perpetrate denial-of-service attacks. With the help of the homemade Thunder Clap program costing just six dollars, Bryan and Anderson [4] leveraged ten virtual servers in Amazon's Elastic Compute Cloud (EC2) system to launch the denial-of-service attacks that succeeded in taking their client company off the Internet. The experiment was not detected by Amazon and no mitigation efforts were initiated against the attacks. The software that controlled the attacks was executed via a command placed on a social network.

Due to the lack of cloud-specific methodologies and tools, traditional digital forensic methodologies and tools are being adapted for use in cloud environments. However, existing digital forensic approaches typically assume that the storage media under investigation are completely under the control of investigators. Thus, these approaches do not map well to cloud computing environments. Cloud computing changes the traditional characteristics of how data – and potential evidence – are stored and retrieved. In the cloud, evidence can reside in different geographical locations on servers that are shared by multiple customers and that are under the control of different cloud service providers. This significantly impacts the identification and acquisition of evidence as well as chain of custody. The fundamental task is to ensure the integrity of evidence retrieved from the cloud so that it may be used in legal proceedings.

2. Traditional Forensics vs. Cloud Forensics

Digital forensics is defined as the use of scientifically-derived and proven methods for the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations [13]. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [12]. Thus, cloud forensics can be defined as the use of proven methods for the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence from distributed computing systems in a manner that maintains the integrity of the evidence so that it is admissible in a court of law.

A digital forensic process model provides a framework for conducting sound forensic investigations. While there is no digital forensic process model that is suited to all digital forensic investigations, a generic process model can be applied to many different types of digital forensic investigations regardless of the technology that is used. This section clarifies the differences between traditional and cloud forensic investigations by engaging the generic Integrated Digital Investigation Process Model proposed by Carrier and Spafford [5]. The model incorporates five phases: (i) preservation; (ii) survey; (iii) search and collection; (iv) reconstruction; and (v) presentation.

2.1 Preservation Phase

The preservation phase of a traditional digital forensic investigation involves securing the digital crime scene and preserving digital evidence. This includes isolating the computer system from the network, collecting volatile data that could be lost when the system is turned off, and identifying any suspicious processes that are running on the system. Suspect users that are logged into the system should be noted and possibly investigated. Log files often contain valuable evidence and should be secured if there is a possibility that they could be lost before the system is copied.

In the case of a cloud forensic investigation, direct physical preservation is limited to the suspect's machine, if this is available. Any other direct preservation is not possible because the data is stored remotely in

virtual images. An investigator can attempt to preserve data resident in the cloud by serving a legal order to a cloud service provider. However, the investigator must trust the service provider to acquire and preserve the data in a forensically-sound manner using proven digital forensic methods.

2.2 Survey Phase

The goal of the survey phase is to identify the obvious pieces of evidence and to develop an initial theory about the incident. Fragile pieces of evidence such as volatile memory are documented and collected immediately to prevent any possible damage or corruption. Carrier and Spafford [5] discuss a server intrusion case in which an investigator looks for obvious signs of a rootkit installation, analyzes application logs and searches for new configuration files. In a cloud environment, the computer system at the scene can be examined for evidence, but the investigator may not have access to external data because the physical examination of remote servers may not possible.

The level to which an investigator can identify potential evidence in a cloud environment is influenced by the specific cloud service model in use – software as a service (SaaS), infrastructure as a service (IaaS), or platform as a service (PaaS). In the SaaS model, the client retains no control over the underlying infrastructure such as the operating system, applications and servers, with the possible exception of limited user-specific application configuration settings. In this case, the investigator has no easy way of identifying evidence on the server side and has to rely on application logs and system logs obtained from the service provider; this is only possible if the service provider has some form of logging mechanism installed and makes the logs available.

The IaaS model offers the most in terms of evidence available to an investigator. In an IaaS environment, the customer controls the set-up of the virtual instances, as well as the underlying operating system and applications. Therefore, the potential exists for customers to install logging applications to keep track of user activity, which could greatly enhance the quality of forensic investigations – but this is not the norm. Nevertheless, an investigator can access more potential evidence than either of the other two cloud service models, SaaS and PaaS.

In the PaaS model, the customer can develop and deploy applications created using programming languages, libraries, services and tools supported by the provider. The customer does not manage or control the underlying cloud infrastructure, including the network, servers, operating systems and storage, but has control over the deployed applications

and possibly the configuration settings for the application-hosting environment [12]. This significantly hinders the ability of an investigator to identify possible evidence, since this is limited to specific application-level logs, if they exist.

Note that documentation is not considered to be a separate phase in a digital forensic investigation because digital evidence is documented as it is found. The final incident report is created during the presentation phase. Digital evidence must be documented thoroughly. For example, a file is documented using its full path, the file system clusters used by the file and the sectors of the disk on which the file resides; the hash value of the file is also computed to ensure that its integrity can be verified. Chain of custody forms must be created if the evidence could be used in court.

Documentation and chain of custody are difficult tasks in a cloud environment. As mentioned above, the level of evidence available to an investigator can vary, which directly affects how well the evidence can be documented. For example, an investigator who has direct access to a virtual image can document the files found on the image. On the other hand, if the investigator relies on the cloud service provider to recover the files of interest, then the investigator has to trust the service provider to retrieve the evidence in a forensically-sound manner.

2.3 Search and Collection Phase

The search and collection phase involves a thorough analysis of the system for digital evidence. This phase uses the results of the survey phase to determine the types of analysis to be performed. For example, a keyword search can be performed during this phase using the keywords identified from other evidence, or a low-level timeline of file activity can be analyzed to trace user activities.

The search and collection phase consumes the bulk of the time spent in an investigation. Artifacts that are of evidentiary value are collected, usually from some type of digital storage device. The collection method involves taking forensic images of the storage devices so that they can be examined under laboratory conditions. Other collection methods are used to retrieve information stored in volatile memory and live registries. The majority of the search and collection phase in a traditional forensic investigation is conducted at the local level, except, for example, the recovery of network logs that typically reside on a server.

The distributed infrastructure of a cloud environment poses challenges with regard to search and collection. The dispersed nature of data in the cloud means that the forensic investigator has to adapt traditional

methods to the new environment. The investigator must understand how data is stored in the cloud environment and determine how it can be retrieved while maintaining its integrity.

At the local level, evidence can be gathered from the client web browser history; this is because communications between the client and cloud service provider typically use an Internet browser. Other evidence, such as client login credentials for cloud services and instant messaging, must also be extracted and deciphered; this can give the investigator access to previous communications conducted by the client over the Internet. At the network level, it is generally not possible to analyze the traffic because service providers may not provide log data from the network components used by the customer's instances and applications.

If the IaaS service model is used, then it is possible for the investigator to take snapshots of the virtual machine and analyze them in a laboratory as in the case of images taken from a local system. The situation is more complex in the case of PaaS because only application-specific data is available. In the case of SaaS, the investigator can only retrieve limited data such as user-specific application configuration settings. The investigator has to provide a court order that would require the provider to execute the search, collect the data and return it to the investigator. The investigator must assume that the service provider employs trustworthy procedures and tools to execute the search, and reassemble and report the data. A violation of the chain of custody can cause the retrieved data to be inadmissible as evidence in court.

2.4 Reconstruction Phase

The reconstruction phase involves organizing the analysis results from the collected physical and digital evidence to develop a theory for the incident. Data that requires advanced analysis techniques, such as executable file analysis or decryption, is processed and the results are used in this phase. Scientific methods are applied to the evidence to test the incident theory. In some cases, the search phase may be resumed again to obtain additional evidence.

In a cloud forensic investigation, the service provider controls the amount of data released to the investigator; the amount of data released affects incident reconstruction. In addition, the physical disparity of the data can make it difficult to put the data in the correct context and temporal order. This situation is exacerbated by the fact that data is held in different geographic regions and the associated computer clocks may not be synchronized. These problems can negatively impact the credibility of the evidence proffered in court.

2.5 Presentation Phase

The presentation phase is the final phase of a forensic investigation. During this phase, all the physical and digital evidence artifacts are documented and presented to court (or to management). Investigator reports, presentations, supporting documentation, declarations, depositions and testimony are considered in the presentation phase. The documentation supporting each phase in the investigation is of particular importance because it helps establish a verifiable chain of custody.

In a forensic investigation, evidentiary data must remain unchanged and the investigator must be competent and able to present the findings, explaining the relevance and implications of all the actions undertaken during the investigation. Furthermore, strict logs and records should be maintained for every step of the investigation. In a cloud environment, it is difficult, if not impossible, to maintain a strict record of an investigation, especially when evidence resides in multiple locations and is under the control of different entities.

2.6 Shortfalls

The discussion above reveals that certain shortfalls during the various phases of the forensic process model can impact a cloud forensic investigation. This could bring into question the validity of the evidence presented in court. The shortfalls are:

- The inability to preserve a potential crime scene, which can adversely affect the integrity of the data artifacts that are collected.
- The unwillingness or inability on the part of the cloud service provider to provide data such as application logs and network logs.
- The limited access or lack of access to cloud data that can provide incomplete pictures of key events.
- The presence of fragmented data and artifacts whose metadata has been altered.

3. Other Issues

Certain other issues related to conducting digital forensic investigations in the cloud can affect the quality of the evidence retrieved. These, in turn, could affect the credibility and admissibility of the recovered artifacts in a court of law.

3.1 Multi-Tenancy

Multi-tenancy allows multiple clients to share a physical server and use services provided by common cloud computing hardware and software simultaneously. In some cases, multi-tenant infrastructures are a concern because the sharing of resources is extensive, occurs at a very large scale and involves multiple potentially vulnerable interfaces [2]. This resource-sharing environment poses challenges to investigators who have to concern themselves not only with the services used by a single customer, but also the non-customer specific components of a multi-tenant infrastructure and the resources shared with other customers. Shared resources include processors and memory. Cloud service providers are often unwilling to give an investigator access to shared memory because it may contain data belonging to other customers and the release of this data could violate confidentiality and privacy agreements.

3.2 Data Provenance

Data provenance records the ownership and process history of data objects and is, therefore, vital to a digital forensic investigation [11]. The provenance can provide information about who or what created the data object and modified its contents. The degree to which data provenance can be implemented in a cloud environment depends on the type of cloud model. In a SaaS implementation, the ancestry of a data artifact may be difficult to trace because the service provider would not normally give an investigator access to application and system log files. In the case of an account compromise, the customer does not have the ability to identify the data that was leaked or accessed by a malicious entity; this includes data modified or deleted by the malicious entity and data deleted by the service provider (e.g., for storage management reasons).

3.3 Multi-Jurisdictional Issues

Data stored in a cloud environment is often distributed over several locations to promote fault tolerance and efficiency of access. However, data distribution raises the issue of jurisdiction, which can present problems in legal proceedings. According to Garrie [7], a court can only hear a matter if it has jurisdiction over the parties and the subject matter of the action. Moreover, law enforcement agencies can only exercise their powers within their authorized jurisdictions.

The problems are exacerbated when data resides in another country. Confidentiality and privacy laws vary greatly from country to country. For example, some countries have strict laws related to the secrecy of

bank documents and the penalties for violating the laws can include criminal sanctions. In such cases, it may not be possible to retrieve all the evidence pertaining to an incident. Garrie [10] cites jurisdictional issues as a major challenge to conducting cloud forensic investigations.

3.4 Chain of Custody

The establishment of a chain of custody is vital to any forensic investigation [1]. It helps provide a documented history of the investigation, detailing "how the evidence was collected, analyzed and preserved in order to be presented as evidence in court" [16].

In a traditional forensic investigation, the chain of custody begins when an investigator preserves evidence at the scene and ends when the evidence is presented in court or to management. The distributed nature of a cloud computing environment significantly complicates the task of maintaining a proper chain of custody. Evidence must be collected from remote servers in a secure and validated manner in order for it to be presented as evidence. If an investigator is unable to gain direct access to cloud services and hardware, it is necessary to rely on the service provider to create forensic copies of evidence. Nevertheless, the investigator must ensure that the chain of custody is always maintained so that cloud data (including data collected by third parties) can be presented as evidence.

3.5 Service Level Agreements

A service level agreement is a contractual document between a cloud service provider and a cloud customer that defines the terms of use of cloud resources. Most current service level agreements do not incorporate provisions regarding forensic investigations and the recovery of evidence from cloud environments. Some provisions regarding the forensic retrieval of evidence from a cloud environment should be incorporated in the agreements. These include data access during forensic investigations and stipulations regarding investigations in multi-jurisdictional and multi-tenant environments, including legal regulations, confidentiality and privacy [15].

At this time, the terms of service are typically prescribed by the cloud provider and are generally non-negotiable [9]. The customer thus has little or no voice regarding the data that the cloud service provider may and may not disclose. Ultimately, the onus is on the customer to negotiate a suitable service level agreement with the provider that addresses evidence retrieval from cloud environments as well as thorny issues such as multiple jurisdictions, data ownership and the establishment of a chain of custody.

4. Cloud Forensic Solutions

Evidently, there are many unresolved issues pertaining to cloud forensic investigations, all of which are exacerbated by the dynamic, ever-changing nature of cloud environments. This section discusses some solutions that could enhance cloud forensic investigations.

4.1 Forensic Tool Testing

Currently, no cloud-specific forensic tool sets are available, requiring investigators to employ tools that were designed for traditional forensic investigations. Evaluation studies focused on the use of existing tools to acquire evidence from cloud environments would immediately benefit the forensics community as well as stimulate forensic tool refinement and new tool development for cloud environments. For example, tools are needed to perform live analyses of dynamic cloud environments. In many cases, live analysis offers the opportunity to gather valuable information from a running system, such as memory and registry data, but no comprehensive solution exists for cloud environments.

Another important gap concerns datasets for testing tools used in cloud forensic investigations. Yet another issue is the correlation of temporal data in cloud forensic investigations. The cloud customer and the service provider often reside in different time zones, which can produce contradicting metadata, such as the creation, modification and last accessed timestamps of an evidence artifact. Methods for automating the correlation of such data would be very beneficial as they would reduce, if not eliminate, the need to conduct intensive manual investigations.

4.2 Transparency of Cloud Services and Data

The lack of transparency regarding the internal infrastructure of a cloud environment poses challenges in an investigation. While information about the internal workings is valuable in an investigation, service providers may provide little information about the environment in which customer data is stored and processed. This lack of transparency is driven by the need to protect sensitive user data; also, releasing information about the internal infrastructure could expose a cloud service to attack [14]. Furthermore, cloud service providers are often unwilling to release information about their environments because it could be used by competitors, and any negative information released about cloud services or operations could harm the reputation of the service provider [3].

Haeberlen [8] proposes that cloud services should be made accountable to the customer and the provider in that both parties should be able to

check whether or not the cloud services are running as agreed upon by both parties. If a problem occurs, the parties should be able to determine which party is responsible and prove the existence of the problem to a third party such as an arbitrator or a judge. This proposal is beneficial to both parties: the customer can check whether or not the contracted services are actually being provided and the service provider can handle complaints and resolve disputes with more ease.

4.3 Service Level Agreements

Service level agreements must include clear and precise procedural information on how a forensic investigation would be handled by the investigator and by the cloud service provider in the event of a criminal incident. The roles should be clearly defined, and each party should be fully aware of its responsibilities, capabilities and limitations. Furthermore, service level agreements must address the legal implications of conducting an investigation in multi-tenant environments across multiple jurisdictions.

4.4 Forensics-as-a-Service

In a forensics-as-a-service (FaaS) model, the cloud service provider should be responsible for forensic data acquisition or, at the very least, provide support for forensic data acquisition. The service provider is in a position to preserve and collect the data because it controls the cloud infrastructure, not only the virtual machines, but also logging and packet capture mechanisms, and billing records. The service could be implemented by a cloud provider with little change to the existing cloud infrastructure, and it would provide customers with the assurance that high-quality forensic investigations could be conducted.

5. Conclusions

Several challenges exist when conducting forensic investigations in cloud environments. These challenges are posed by the highly dynamic, distributed, multi-jurisdictional and multi-tenant nature of cloud environments. Failure to address these challenges could affect the credibility and admissibility of the recovered digital evidence. Promising solutions include the development of cloud-ready forensic tools and service level agreements with built-in provisions for forensic investigations. However, the most complete solution would be to ensure that service providers implement forensics-as-a-service (FaaS) as a standard offering. This would enable high-quality forensic investigations to be conducted using traditional digital forensic tools under existing service level agreements.

References

[1] Association of Chief Police Officers, Good Practice Guide for Computer-Based Evidence, London, United Kingdom, 2012.

[2] L. Badger, R. Bohn, S. Chu, M. Hogan, F. Liu, V. Kaufmann, J. Mao, J. Messina, K. Mills, A. Sokol, J. Tong, F. Whiteside and D. Leaf, U.S. Government Cloud Computing Technology Roadmap, Volume II, Release 1.0 (Draft), Useful Information for Cloud Adopters, NIST Special Publication 500-293, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[3] D. Birk and C. Wegener, Technical issues of forensic investigations in cloud computing environments, *Proceedings of the Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, 2011.

[4] D. Bryan and M. Anderson, Cloud computing, A weapon of mass destruction? presented at the *DEFCON 18 Hacking Conference*, 2010.

[5] B. Carrier and E. Spafford, Getting physical with the digital investigation process, *International Journal of Digital Evidence*, vol. 2(2), 2003.

[6] S. Garfinkel, The criminal cloud, *MIT Technology Review*, October 17, 2011.

[7] D. Garrie, Cloud computing and jurisdiction, Part 2: A primer, Law and Forensics, Seattle, Washington (`www.lawandforensics.com/cloud-computing-jurisdiction-part-primer`), 2012.

[8] A. Haeberlen, A case for the accountable cloud, *ACM SIGOPS Operating Systems Review*, vol. 44(2), pp. 52–57, 2010.

[9] W. Jansen and T. Grance, Guidelines on Security and Privacy in Public Cloud Computing, NIST Special Publication 800-144, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[10] S. Liles, M. Rogers and M. Hoebich, A survey of the legal issues facing digital forensic experts, in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 267–276, 2009.

[11] R. Lu, X. Lin, X. Liang and X. Shen, Secure provenance: The essential of bread and butter of data forensics in cloud computing, *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, pp. 282-292, 2010.

[12] P. Mell and T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[13] G. Palmer, A Road Map for Digital Forensic Research – Report from the First Digital Forensic Research Workshop, DFRWS Technical Report, DTR-T001-01 FINAL, Air Force Research Laboratory, Rome, New York, 2001.

[14] T. Ristenpart, E. Tromer, H. Schacham and S. Savage, Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds, *Proceedings of the Sixteenth ACM Conference on Computer and Communications Security*, pp 199–212, 2009.

[15] K. Ruan, J Carthy, T. Kechadi and M. Crosbie, Cloud forensics, in *Advances in Digital Forensics VII*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp 35–46, 2011.

[16] J. Vacca, *Computer Forensics: Computer Crime Scene Investigation*, Charles River Media, Hingham, Massachusetts, 2002.

Chapter 21

RULE-BASED INTEGRITY CHECKING OF INTERRUPT DESCRIPTOR TABLES IN CLOUD ENVIRONMENTS

Irfan Ahmed, Aleksandar Zoranic, Salman Javaid, Golden Richard III and Vassil Roussev

Abstract An interrupt descriptor table (IDT) is used by a processor to transfer the execution of a program to software routines that handle interrupts raised during the normal course of operation or to signal an exceptional condition such as a hardware failure. Attackers frequently modify IDT pointers to execute malicious code. This paper describes the IDTchecker tool, which uses a rule-based approach to check the integrity of the IDT and the corresponding interrupt handling code based on a common scenario encountered in cloud environments. In this scenario, multiple virtual machines (VMs) run the same version of an operating system kernel, which implies that IDT-related code should also be identical across the pool of VMs. IDTchecker leverages this scenario to compare the IDTs and the corresponding interrupt handlers across the VMs for inconsistencies based on a pre-defined set of rules. Experimental results related to the effectiveness and runtime performance of IDTchecker are presented. The results demonstrate that IDTchecker can detect IDT and interrupt handling code modifications without much impact on guest VM resources.

Keywords: Cloud forensics, interrupt descriptor table, integrity checking

1. Introduction

Memory forensics involves the extraction of digital artifacts from the physical memory of a computer system. The interrupt descriptor table (IDT) is a valuable artifact, more so because it is a well-known target for malware (especially rootkits). The IDT provides an efficient way to transfer control from a program to an interrupt handler, a special software routine that handles exceptional conditions occurring within

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 305–328, 2013.

the system, processor or executing program. (A hardware failure and division by zero are examples of unusual conditions that are handled by an IDT.) Malware often manipulates an IDT to change the system control flow and run malicious code. The change may occur either to a pointer in the IDT or in the interrupt handler itself, which then redirects execution to malicious code that has been injected into the system.

PatchGuard [6, 18] checks IDT integrity by keeping a valid state of the table and comparing it with the current table state. Microsoft includes kernel patch protection (PatchGuard) [18] in 64-bit Windows systems to detect modifications to kernel code and critical data structures such as the IDT. PatchGuard caches the legitimate copy and the checksum of the IDT, and compares them with the current IDT in memory to check for modifications. Another tool, CheckIDT [6], examines IDT integrity by storing the entire table in a file so that it can be compared later with the current state of the table in memory.

While these tools are applicable to different operating systems, they suffer from at least two major limitations. Both tools require an initialization phase, where it is assumed that the IDT is not infected at the time that the valid state of the IDT is obtained. This may not be the case if the interrupt handler is patched in the kernel file on disk because, when the system restarts and the IDT is created, the IDT points to the malicious interrupt handler before the valid state of the IDT can be obtained. The pointers in the IDT may change after the system boots if the kernel or kernel modules are loaded at different memory locations. Thus, every time the system restarts, the tools need to record the valid state of the table.

Current solutions do not specifically consider the interrupt handler code for integrity checking. Although they do check the integrity of the kernel code and modules that include interrupt handler code, they do not ensure that the pointer in the IDT points to a valid interrupt handler. A state-of-the-art solution for checking the integrity of kernel code (and modules) requires the maintenance of a dictionary of cryptographic hashes of trusted code [9, 14] in order to compare the hash of the current code with the hash stored in the dictionary. Such an approach requires maintaining the dictionary across every kernel update to implement effective integrity checking.

This paper describes the IDTchecker tool, which provides a comprehensive, rule-based approach to check IDT integrity in real time without requiring an initialization phase, a "known-good" copy of the IDT or a dictionary of hashes. IDTchecker works in a virtualized environment where a pool of virtual machines (VMs) run identical guest operating systems with the same kernel version – a typical scenario in cloud

servers. The pools of virtual machines simplify the maintenance process to facilitate the automation of patch applications and system upgrades. IDTchecker works by retrieving the IDT and its corresponding interrupt handler code from the physical memory of a guest VM and comparing it across the VMs in the pool. The tool uses a pre-defined set of rules to perform comprehensive integrity checking. It runs on a privileged virtual machine where it has access to guest VM physical memory through virtual machine introspection (VMI). None of the components of IDTchecker run inside the guest VMs, which makes IDTchecker more resistant to tampering by malware.

IDTchecker was evaluated extensively in order to assess its effectiveness and efficiency. Effectiveness testing used real-world malware and popular IDT exploitation techniques to modify the IDT and interrupt handler code. Efficiency testing analyzed the runtime performance of IDTchecker under the best case and worst case scenarios. The results demonstrate that IDTchecker does not have any significant impact on guest VM resources because none of its components run inside the VMs. Its memory footprint is 10 to 15 MB, which is quite negligible compared with the amount of physical memory typically found in a cloud server (tens to hundreds of gigabytes).

2. Related Work

This section discusses research related to IDT integrity checking.

CheckIDT [6] is a Linux-based tool that detects IDT modifications by storing the IDT descriptor values in a file and later comparing them with the current values of the IDT in memory. If a discrepancy between the two tables is detected, CheckIDT restores the table in memory by copying the IDT values from the saved file, assuming that the integrity of the file has been maintained. CheckIDT uses the technique described in [16] to access the table from user space without using a Linux kernel module.

Kernel Patch Protection [18] (or PatchGuard) checks the integrity of kernel code (including modules) and important data structures such as the IDT, global descriptor table (GDT) and system service descriptor table (SSDT). It is currently implemented in 64-bit Windows operating systems. PatchGuard stores legitimate known-good copies and checksums of kernel code and data structures, and compares them with the current state of the code and the data structures at random times. PatchGuard is implemented as a set of routines that are protected by using anonymization techniques such as misdirection, misnamed functions and general code obfuscation.

Volatility [21] has a plugin [22] that checks the integrity of IDT pointers to interrupt handlers. It walks through each IDT entry and checks if the pointer in the entry is within the address range of the kernel code (including modules). Volatility ensures that the pointers do not point to unusual locations in memory. However, it cannot detect attacks [6, 10] that directly patch interrupt handler code and do not modify pointers in the table.

IDTGuard [19] is a Windows-based tool that checks the integrity of IDT pointers. The tool separately computes an IDT pointer value by finding the offset of the interrupt handler in the kernel file (such as `ntoskrnl.exe`) and adding it to the base address of the kernel in memory. The computed IDT pointer values are then matched with the pointers in the IDT table to verify their integrity. However, IDTGuard cannot check the integrity of pointers corresponding to kernel modules where the pointers point to `Kinterrupt` data structures instead of interrupt handlers in the kernel code.

3. IDT Integrity Checking

Virtualization provides an opportunity for efficient resource utilization of a physical machine by concurrently running several VMs over a virtual machine monitor (VMM) or hypervisor – an additional layer between the hardware and the hosted guest operating systems. The VMM also allows a privileged VM to monitor the runtime resources of other (guest) VMs (e.g., memory and I/O) through virtual machine introspection. IDTchecker uses introspection while running on a privileged VM to access the physical memory of guest VMs. It retrieves the IDTs and their corresponding interrupt handlers and matches them according to pre-defined rules in order to check for inconsistencies.

3.1 Overview

Interrupts and exceptions are system events that indicate that a condition or an event requires the attention of the processor [5]. Interrupts can be generated in response to hardware signals such as hardware failure or by software through the `INT n` instruction. Exceptions are generated when the processor detects an error during the execution of an instruction such as divide by zero. Each condition indicated by an interrupt or exception requires special handling by the processor and, thus, is represented using a unique identification number referred to as an interrupt vector. In this paper, the difference between interrupts and exceptions is not important and, thus, both are referred to as interrupts. When an in-

terrupt occurs, the execution of a program is suspended and the control flow is redirected to an interrupt handler routine through an IDT.

An IDT is an array of interrupt vectors, each vector providing an entry point to an interrupt handler. There can be at most 256 interrupt vectors. Each vector is eight bytes long and contains information about the index to a local/global descriptor table, request/descriptor privilege levels, offset to interrupt handler, etc. There are up to three types of interrupt (vector) descriptors in an IDT: interrupt gate, trap gate and task gate. Interrupt and trap gate descriptors are similar, but they differ in functionality and in the type field in the descriptor that identifies the gate. Unlike the situation for trap gates, when handling an interrupt gate, the processor clears the IF flag in the EFLAGS register to prevent other interrupts from interfering with the current interrupt handler. Task gate descriptors, on the other hand, have no offset values to the interrupt handler. Instead, the interrupt handler is reached through the segment selector field in the descriptor.

The global descriptor table (GDT) is utilized when the interrupt handler has to be accessed in protected mode (where a protection ring [5] is enforced). An entry in the table is called a segment descriptor. Each descriptor describes the base address and the size of a memory segment along with information related to the access rights of the segment. Each descriptor is also associated with a segment selector that provides information about the index to the descriptor, access rights and a flag that determines if the index points to an entry of the global descriptor table. Each interrupt vector has a segment selector that is used to find the base address of the segment. In a case of interrupt and trap gates, the base address of the interrupt is obtained by adding the base address of the segment to the offset in the vector.

3.2 Assumptions

We assume the presence of a fully-virtualized environment where the VMM supports memory introspection of guest VMs. Also, we assume that different pools of VMs are present, where each pool runs an identical guest operating system with the same kernel version. This provides an opportunity for IDTchecker to probe and compare the IDTs and interrupt handlers within each pool.

3.3 IDTchecker Architecture

IDTchecker is designed to obtain the IDTs and corresponding interrupt handler code from a pool of VMs and perform a comprehensive integrity check based on a pre-defined set of rules. To achieve this

task, IDTchecker employs four modules to implement the various functions needed to perform integrity checking. The four components are: (i) Table-Extractor; (ii) Code-Extractor; (iii) Info-Extractor; and (iv) Integrity-Checker.

- **Table-Extractor and Code-Extractor:** IDTchecker incorporates separate modules (Table-Extractor and Code-Extractor) for extracting tables and interrupt related code because the IDT and GDT structures are dependent on the processor, while the organization of the interrupt-related code is mostly dependent on the operating system. For instance, Microsoft Windows uses a `Kinterrupt` structure to store the information about an interrupt handler that is provided by kernel drivers. Separating the extraction of code and tables into two modules increases the portability of IDTchecker. Moreover, Code-Extractor receives the interrupt vector descriptor values from Table-Extractor after the descriptors are parsed. This data is used by Code-Extractor to locate the index of a GDT segment and the offset of an interrupt handler (if the descriptor type is not a task gate).

- **Info-Extractor:** The Info-Extractor module fetches any additional information required by a rule from memory, such as the address range of kernel modules.

- **Integrity-Checker:** The Integrity-Checker module applies a predefined set of rules to the data obtained from the Table-Extractor, Code-Extractor and Info-Extractor modules in order to comprehensively check IDT integrity. Unlike the other three modules, Integrity-Checker does not need to access the memory of guest VMs. This is because all the needed data is made available by the other modules.

Figure 1 presents the overall architecture of IDTchecker. The figure shows multiple pools of guest VMs, each pool running the same version of a guest operating system. The VMs run on top of a VMM and the VMI facility is available for a privileged VM to introspect guest VM resources. Note that IDTchecker only needs to perform read-only operations on guest VM physical memory and no IDTchecker component runs inside a guest VM.

By comparing IDTs and their corresponding interrupt handler code across VMs, IDTchecker is able to detect IDT inconsistencies between the VMs. A majority vote algorithm is used to identify an infected VM. Of course, the majority vote strategy is effective only if the majority of VMs have uninfected IDTs. In this case, IDTchecker is more effective

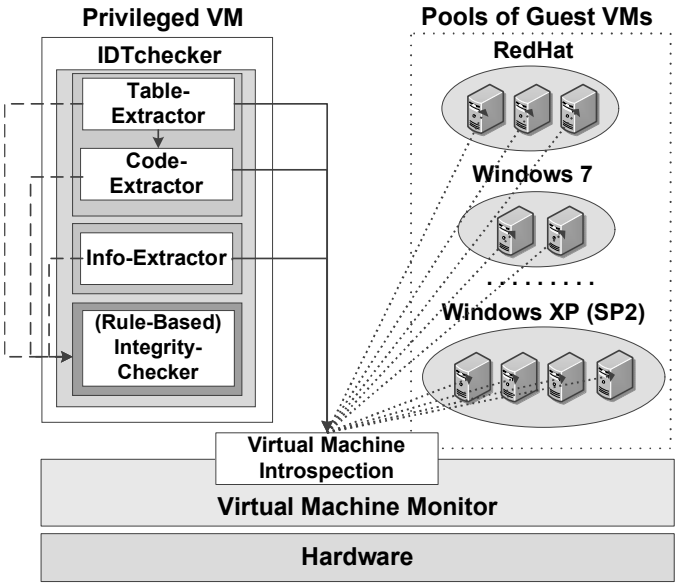


Figure 1. IDTchecker architecture.

at detecting the first sign of infection, which can then be used to trigger a thorough forensic investigation to find the root cause of the infection.

It is also worth discussing if the IDT should always be identical across VMs when identical kernel code (including modules) is executing. The initial 32 interrupt vectors (0 to 31) are pre-defined. These interrupt vectors always remain identical across VMs. However, other interrupt vectors (32 to 255) are user-defined and may vary across VMs in that the same interrupt entry or descriptor can be associated with different interrupt vectors across VMs. Thus, one-to-one matching of interrupt entries may not be feasible at all times. A more robust approach is to find the equivalent interrupt vector entry that is being matched in other IDT tables across VMs before the rules are applied to them. The current version of IDTchecker performs one-to-one matching and can be enhanced using this approach.

3.4 Integrity Checking Rules

IDTchecker currently uses four rules to perform integrity checking across VMs and within each VM:

- **Rule 1:** All the values in each interrupt vector should be the same across VMs (excluding the interrupt handler offset field, which is

checked for integrity by the subsequent rules). This rule ensures that all the fields in the IDT are original.

- **Rule 2:** The interrupt handler code should be consistent across VMs. This rule detects modifications to the code. The rule is effective unless identical modifications are made to the code in all the VMs.
- **Rule 3:** The interrupt handler is located in basic kernel code or in a kernel module. This means that the base address of the interrupt handler should be within the address range of the basic kernel code or within the address range of the code of a module. This rule ensures that the base address does not point to an unusual location.
- **Rule 4:** Given that the base address of an interrupt handler is within the address range of the kernel code or of a module, the offset of the base address of the interrupt handler from the starting address of its corresponding driver or basic kernel code should be the same across all VMs. This rule detects instances of random injections of malicious code within the basic kernel code or within a driver.

4. IDTchecker Implementation

The IDTchecker design is simple in that all its components reside locally on a privileged VM, which can be implemented on any VMM that has memory introspection support (e.g., Xen, KVM or VMware ESX) without requiring modifications to the VMM itself. For the proof of concept, we developed IDTchecker on Xen [23] with the Microsoft Windows (Service Pack 2) XP guest operating system. We used the LibVMI introspection library [20] and the Opdis disassembler library [12]; cryptographic hashes were computed using OpenSSL [13].

The remainder of this section describes the low-level implementation details of the IDTchecker components.

4.1 Table-Extractor

The IDT and GDT are created each time a system starts. The processor stores their base addresses and sizes in `IDTR` and `GDTR` registers for protected mode operations. In each register, the base address specifies the linear address of byte `0` of the table, while the size specifies the number of bytes in the table. Table-Extractor obtains this information from the registers in the guest VM and extracts the IDT and GDT tables from the guest VM memory. It further interprets the raw bytes of

the tables as table entries and their respective fields, and forwards them to Code-Extractor.

4.2 Code-Extractor

Code-Extractor receives the tables from Table-Extractor and retrieves the code corresponding to each IDT entry. Code-Extractor handles each interrupt vector type (interrupt gate, task gate and trap gate) differently. Since no trap gate entries are found in Windows XP VMs, only the interrupt and task gate extraction are discussed below.

4.2.1 Interrupt Gate. Each interrupt gate entry in the IDT has a segment selector associated with the GDT. It also has an interrupt handler offset that can be added to the base address of the segment described in the GDT to form the base address `ptr` of the interrupt handler. The interrupt handler can be located in the basic kernel code (i.e., `ntoskrnl.exe` for Windows XP) or in a kernel module. If the interrupt handler is in a kernel module, then `ptr` points to the `Kinterrupt` data structure, which is a kernel control object that allows device drivers to register an interrupt handler for their devices. The data structure contains information that the kernel needs to associate the interrupt handler with a particular interrupt (e.g., the base address of the interrupt handler in the module and the vector number of the IDT entry). In order to determine if the handler code is in a kernel module, the vector number in the `Kinterrupt` structure is matched with the vector number of the IDT entry. If the two values match, the handler code is in the kernel module, otherwise it is in the basic kernel.

At this stage, Code-Extractor needs to find the base addresses and sizes of all the interrupt handling code segments in order to make a clean extraction of the code. Figure 2 shows the extraction process. Note that the IDT and GDT descriptor formats in the figure are adjusted for illustrative purposes.

- **Finding the Base Address:** When the code is located in the basic kernel, `ptr` contains the base address of the interrupt handler, which is the only code needed for integrity checking. When the code is in a module, `ptr` points to `DispatchCode`, which executes and at some point jumps to other code (`InterruptDispatcher`). This code executes and at some point calls the interrupt handler from the device driver. In this case, three chunks of code have to be extracted. The base addresses of the three pieces of code are in the `Kinterrupt` structure, which Code-Extractor processes to obtain the addresses.

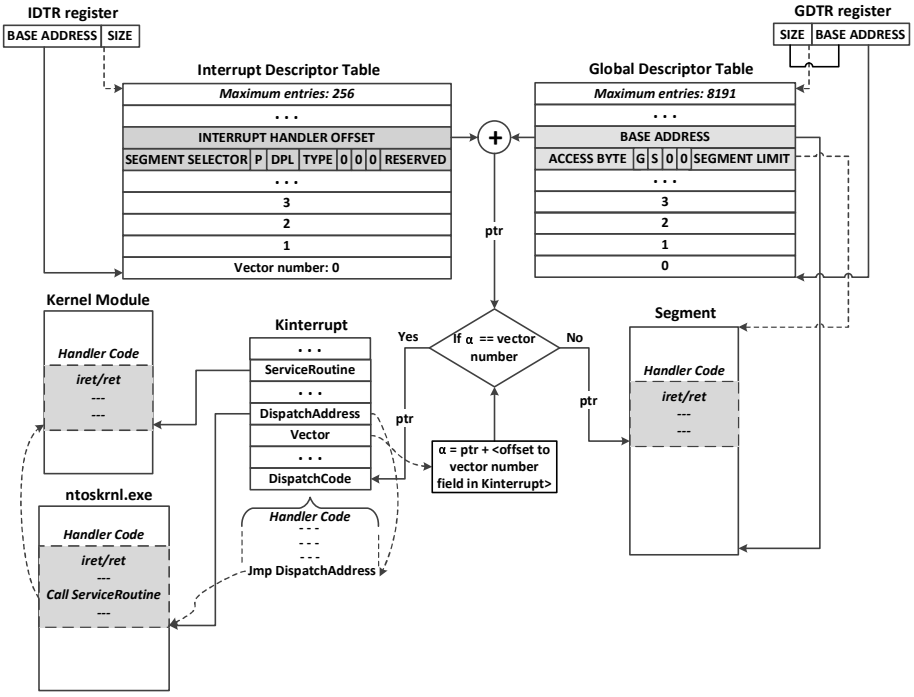


Figure 2. Code extraction of an interrupt gate.

- **Finding the Code Size:** Code-Extractor finds the size of the code by disassembling it starting from the base address of the code, assuming that the first occurrence of a return instruction points to the end of the code. This assumption is valid based on the function prologue and epilogue convention, which is followed by assembly language programmers and high-level language compilers. The function prologue and epilogue are small amounts of code placed at the start and end of a function, respectively; the prologue stores the state of the stack and registers when the function is called and the epilogue restores them when the function returns. Thus, a return instruction is required at the end of a function in order to ensure that the restoration code executes before function returns.

  We have not encountered a situation where a return in interrupt handler code occurs before the end of the handler. Also, we performed an experiment to see if the Windows Driver Model (WDM) compiler follows the function prologue and epilogue convention. We placed a few return instructions between the if-else statements in the interrupt handler code of a hello-world driver. After compiling the code, we discovered that the return instructions were

replaced with jump instructions pointing to return instructions placed at the end of the code. This shows that the WDM compiler follows the convention upon which our heuristic relies.

4.2.2 Task Gate. Each task gate entry in the IDT has no interrupt handler offset and, therefore, there is no direct pointer to a handler or code. Instead, the segment selector in the entry is the index of a GDT entry. The GDT entry is a task state segment (TSS) descriptor that provides information about the base address and size (i.e., segment limit) of a TSS. The TSS stores the processor state information (e.g., segment registers and general purpose registers) that is required to execute the task. The TSS also contains the code segment (`CS`) that points to one of the descriptors in the GDT that defines a segment where the interrupt handler code is located. Additionally, the TSS contains the instruction pointer (`EIP`) value. When a task is dispatched for execution, the information in the TSS is loaded into the processor and task execution begins with the instruction pointer (`EIP`) value, which provides the base address of the interrupt handler.

After locating the base address of the interrupt handler, the process for determining the code size is the same as that used for interrupt gates. Figure 3 shows the extraction process. Note that the IDT and GDT descriptor formats in the figure are adjusted for illustrative purposes.

4.3 Info-Extractor

The Info-Extractor module is used to obtain additional information associated with the IDT and its code and make it available to Integrity-Checker. In addition, the module obtains the address range of the basic kernel and its associated modules that Integrity-Checker requires for Rules 3 and 4. Info-Extractor also takes into consideration the other modules that are already loaded in memory.

Windows XP maintains a doubly-linked list (Figure 4) corresponding to the locations of the basic kernel code and modules, where each element in the list is a `LDR_DATA_TABLE_ENTRY` data structure that contains the base address `DllBase` and the size of the module `SizeOfImage`. Windows XP also stores the pointer to the first element of the list in a system variable `PsLoadedModuleList`, which Info-Extractor uses to reach the list, browse each element and store it in a local buffer. The pointer to the buffer is then forwarded to Integrity-Checker.

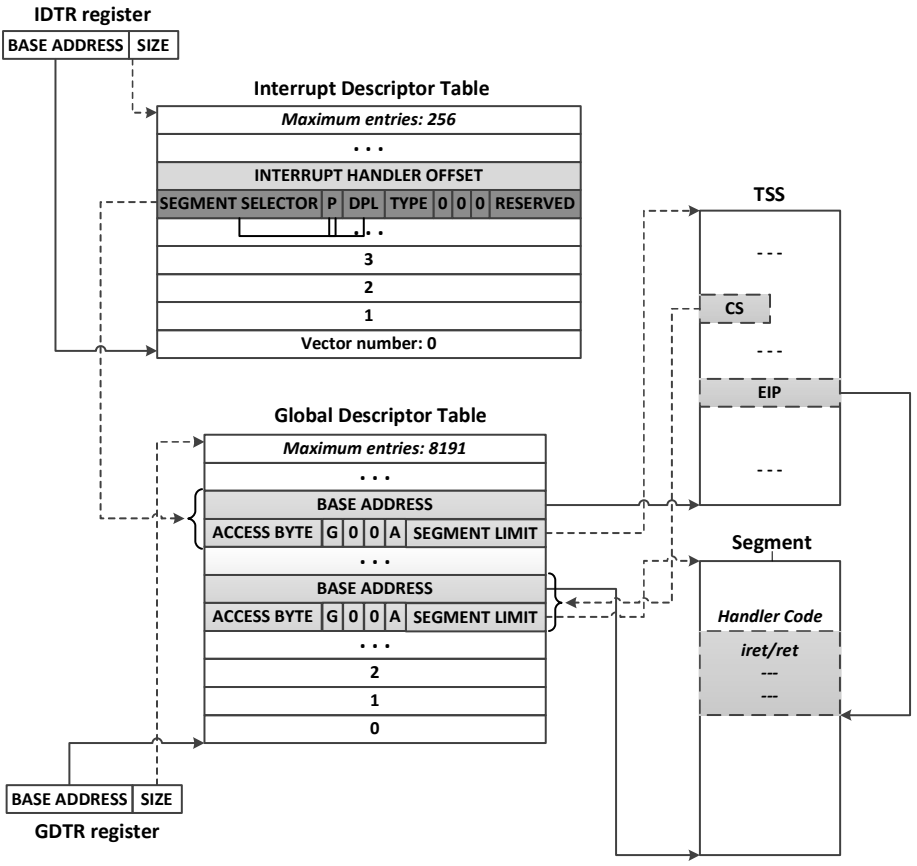


Figure 3. Code extraction of a task gate.

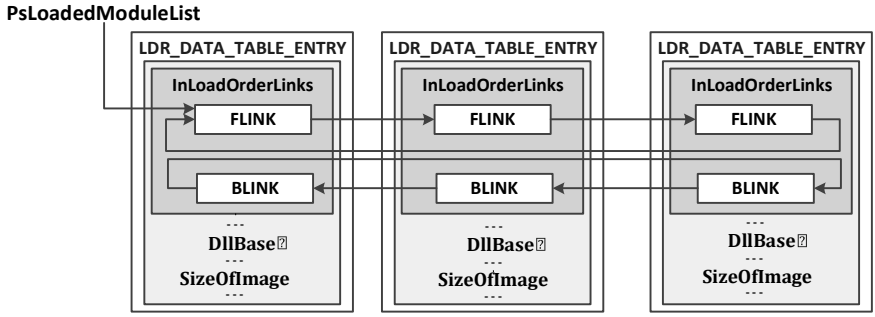


Figure 4. Doubly-linked list of kernel modules

4.4 Integrity-Checker

Integrity-Checker applies the rules to the data obtained from the other three modules. However, Integrity-Checker sometimes needs to also manipulate the data in order to apply the rules. We discusses both these aspects of Integrity-Checker with regard to each rule.

- **Rule 1:** This rule compares IDTs across VMs. Integrity-Checker does this by comparing each value of every IDT entry across VMs. However, this does not include interrupt handler offsets.

- **Rule 2:** This rule compares the interrupt handler code across VMs. Integrity-Checker uses the interrupt handler code obtained by Code-Extractor. However, because the code has been extracted from the memory of different VMs, it may not always match. The reason is that the code of the basic kernel and its modules in the files have relative virtual addresses (RVAs) or offsets. When a module is loaded into memory, the loader replaces the RVAs with absolute addresses by adding the base address of the module (i.e., the pointer to the zeroth byte of the module in memory) to the RVAs. If the same module is loaded at different locations across VMs, the kernel/module code (including the interrupt handler in the code) will have different absolute addresses and, as a result, will not be consistent and will not match. Integrity-Checker reverses this change by subtracting the base addresses of the modules from the absolute addresses in the code. This brings the absolute addresses back to RVAs, which represent the values in files and should be the same across VMs.

  Figure 5 illustrates the RVA modification of the interrupt handler `i8042prt!I8042KeyboardInterruptService` associated with interrupt vector `0x31`. The 32-bit base addresses of the module for virtual machines VM1 and VM2 are `F8 7B A4 95` and `F8 7D A4 95`, respectively. The figure shows the same interrupt handler code extracted from the physical memory of the two VMs. Integrity-Checker assumes that the differences of bytes in the code represent the absolute addresses. This assumption is valid until the code in one of the VMs is modified because the base address of the module containing the handler is different for the two VMs. Ideally, there should be a difference of four bytes for a 32-bit machine. However, depending on where the difference of bytes starts in the base address of the module in the two VMs, this may not always be the case.

```
00000000|  6a 18 68 a8 d7 7d f8  e8  ff 00 00 00 8b 7d 0c 8b   j.h..}.......}..
00000010|  77 28 83 7e 30 01 0f 85  4f 01 00 00 a1 00 d9 7d   w(.~0...O......}
00000020|  f8  ff b0 a4 00 00 00 ff  15 0c d9 7d f8 88 45 df   ...........}..E.
00000030|  24 21 33 db 3c 01 0f 85  f3 19 00 00 8d 45 e3 50   $!3.<........E.P
00000040|  6a 01 e8 18 ff ff ff 8d  86 4a 01 00 00 8a 08 88   j........J......
                              . . .
                              . . .
00000210|  0f 85 ed 00 00 00 e9 b1  00 00 00 8b 08 8b 91 f4   ................
00000220|  01 00 00 89 50 08 8b                               ....P..
MD5: fcd7298fa2a2f3f606c997ecd8c90392
```

(a) VM1 before RVA modification.

```
00000000|  6a 18 68 a8 d7 7b f8  e8  ff 00 00 00 8b 7d 0c 8b   j.h..{.......}..
00000010|  77 28 83 7e 30 01 0f 85  4f 01 00 00 a1 00 d9 7b   w(.~0...O......{
00000020|  f8  ff b0 a4 00 00 00 ff  15 0c d9 7b f8 88 45 df   ...........{..E.
00000030|  24 21 33 db 3c 01 0f 85  f3 19 00 00 8d 45 e3 50   $!3.<........E.P
00000040|  6a 01 e8 18 ff ff ff 8d  86 4a 01 00 00 8a 08 88   j........J......
                              . . .
                              . . .
00000210|  0f 85 ed 00 00 00 e9 b1  00 00 00 8b 08 8b 91 f4   ................
00000220|  01 00 00 89 50 08 8b                               ....P..
MD5: 5e87703b1a42456c4928b6cc60b8ea96
```

(b) VM2 before RVA modification.

```
00000000|  6a 18 68 13 33 00 00  e8  ff 00 00 00 8b 7d 0c 8b   j.h..}.......}..
00000010|  77 28 83 7e 30 01 0f 85  4f 01 00 00 a1 6b 34 00   w(.~0...O......}
00000020|  00  ff b0 a4 00 00 00 ff  15 77 34 00 00 88 45 df   ...........}..E.
00000030|  24 21 33 db 3c 01 0f 85  f3 19 00 00 8d 45 e3 50   $!3.<........E.P
00000040|  6a 01 e8 18 ff ff ff 8d  86 4a 01 00 00 8a 08 88   j........J......
                              . . .
                              . . .
00000210|  0f 85 ed 00 00 00 e9 b1  00 00 00 8b 08 8b 91 f4   ................
00000220|  01 00 00 89 50 08 8b                               ....P..
MD5: 3925130249749612de2cbd3fc8a6182b
```

(c) VM1 after RVA modification.

```
00000000|  6a 18 68 13 33 00 00  e8  ff 00 00 00 8b 7d 0c 8b   j.h..}.......}..
00000010|  77 28 83 7e 30 01 0f 85  4f 01 00 00 a1 6b 34 00   w(.~0...O......}
00000020|  00  ff b0 a4 00 00 00 ff  15 77 34 00 00 88 45 df   ...........}..E.
00000030|  24 21 33 db 3c 01 0f 85  f3 19 00 00 8d 45 e3 50   $!3.<........E.P
00000040|  6a 01 e8 18 ff ff ff 8d  86 4a 01 00 00 8a 08 88   j........J......
                              . . .
                              . . .
00000210|  0f 85 ed 00 00 00 e9 b1  00 00 00 8b 08 8b 91 f4   ................
00000220|  01 00 00 89 50 08 8b                               ....P..
MD5: 3925130249749612de2cbd3fc8a6182b
```

(d) VM2 after RVA modification.

Figure 5. VM1 and VM2 before and after RVA modification.

Currently, Integrity-Checker considers only the interrupt handler code for integrity checking, which is sufficient unless the routines called by the handler are patched with malicious code. In this case, IDT integrity is violated although the IDT table and its related interrupt handler code are still intact. Instead of finding such routines and checking their integrity, it is more efficient to check the integrity of the entire module where the handler code is located.

This may also include the routines that are not being called, but this approach can reduce the time required to search for the calling functions. Several techniques have been proposed for checking the integrity of entire modules (see, e.g., [1, 4, 7–9, 15, 17]).

- **Rules 3 and 4:** Rules 3 and 4 check the base address of the interrupt handler code in the address range of the kernel modules and check the offset of the handler base address from the base address of its respective module. Integrity-Checker has the list of kernel modules and their address ranges (where the address ranges are exclusive and do not overlap). Integrity-Checker searches the base address of interrupt handler to check if it is within the address range of a module. It uses a binary search that requires the list associated with a module to be sorted according to the base address. When the handler base address is found in the address range of a module, the module is considered to be a holder of the interrupt handler. The module base address is also used to compute the offset between the base addresses of the module and the handler, which is then matched across VMs.

5. Evaluation

This section presents the results of several experiments that evaluated the effectiveness and efficiency of IDTchecker.

5.1 Experimental Setup

We built a small-scale cloud server for the experiments. The server ran Xen 4.1.3 on an Intel Core 2 Quad (4 × 2.83 GHz cores) with 8 GB RAM. We created seven VMs (i.e., DomUs) using Xen. Each VM had 1 GB RAM, a 10 GB hard disk and ran Windows XP (Service Pack 2) using hardware-assisted virtualization. The privileged VM (i.e., Dom0) ran Fedora 16 with the `3.4.9-2.fc16.x86_64` kernel.

5.2 Integrity Checking

IDTchecker is designed to detect integrity violations in the IDT and its corresponding interrupt handlers. This section presents the results of experiments that violated IDT integrity in various ways. The experiments employed real-world malware to manipulate the IDT and execute malicious code.

5.2.1 Hooking an Interrupt. Each IDT descriptor has a 32-bit pointer that points to an interrupt handler or the `Kinterrupt`

structure. Each pointer is formed from the two 16-bit fields in the descriptor, which are the lower and higher 16 bits of the pointer address. Techniques are available to exploit this pointer to redirect control flow to malicious code [6].

This experiment modified the IDT pointer and tested if IDTchecker could detect the modification. An implicit malfunctioning behavior of the IDTGuard tool [19] was used to effect the modification. As discussed in Section 2, this tool is designed to check IDT integrity by separately computing the pointer values and comparing them with the values in the IDT. However, the computation is only possible when the interrupt handlers are located in the kernel code (i.e., `ntoskrnl.exe`). When IDTGuard computes the value of the pointer to `Kinterrupt` (because the interrupt handler is in kernel module), it computes a pointer value of a random location in a kernel code. Thus, we used IDTGuard to replace the original pointer value in the IDT with the random pointer value. IDTchecker was able to detect the modification by showing that the code pointed to by the pointer was different from the code pointed to by the pointers in the other VMs.

5.2.2 Hooking an Interrupt Handler . An interrupt handler can also be patched in order to run malicious code [6]. This change would not modify the pointer in the IDT, but the actual code that is executed to handle an interrupt. This experiment used a customized driver for a programmed I/O device [11] with an interrupt handler. When the driver was loaded, the interrupt handler was registered with an interrupt vector. We used an IDT entry (`0x3e`) that was originally registered with the `atapi!IdePortInterrupt` handler. Next, we disabled the IDE channel to free system resources to hold the programmed I/O device [11]. We then installed the driver for this device, which also registered the interrupt handler with vector `0x3e`. IDTchecker detected the modification by comparing the IDTs across the other VMs and showing that the interrupt handler code for vector `0x3e` was different from the corresponding code in the other VMs.

5.2.3 IDT Manipulation via Malware. Real-world malware and IDT exploitation techniques can modify IDT pointers and interrupt handler code in order to run malicious code. Three experiments were conducted to test the performance of IDTchecker in the face of IDT manipulation via malware.

- **Subverting the Windows Kernel:** As discussed by Skape [18], rootkits that directly replace IDT entries leave many traces and are, therefore, not stealthy. Rootkits that are largely undetectable

by common scanners rely on overwriting an interrupt handler such as the `KiInterruptTemplate` routine pointed to by the interrupt vector. mxatone and ivanlef0u [10] have demonstrated how to attach keylogging or packet sniffing code via IDT hooking. Their technique searches for the code `"mov edi, <&Kinterrupt>; jmp edi;"` and modifies the pointer in `KiInterruptTemplate` to point to the maliciously crafted `Kinterrupt` structure that contains calls to the kernel routines that can gather keyboard strokes or network packets. The original interrupt handler will still execute after the malicious interrupt handler because the malicious code returns to the legitimate `Kinterrupt` structure. After modifications were made to the `Kinterrupt` structure, IDTchecker was able to detect the code injection by `KinterruptTemplate` pointer modification.

- **Direct Kernel Hooking:** A proof-of-concept malware [2] registers a dummy driver that hooks interrupt vectors `0x01` and `0x03` to functions that represent a USB storage device as a regular disk drive. This is done by capturing calls to `IoCreateDevice()` that take a pointer to the `DRIVER_OBJECT` of the newly-added device and replaces the MajorFunction (`IRP_MJ_DEVICE_ CONTROL`) with the malicious function sitting in the dummy driver. As a result, every system call to the USB device driver `USBSTOR` can be intercepted and monitored. In order to do so, the malware hooks `IoCreateDevice()` by inserting instructions into the executable code. An IDT hooking function contained within the dummy driver is called directly from the dummy driver. The `hookIDT()` calling function preserves the old interrupts (`0x01` and `0x03`) that are to be hooked. After the original IDT has been preserved, the hooking mechanism can be unleashed, which hooks the debugging interrupts `0x01` and `0x03`. After this is done, the original IDT is restored and the addresses of newly-created hooks are added to the list of hooked IDT entries.

  IDTchecker ran a comparative analysis of two VMs, where one of the VMs (VM1) had registered the malicious driver. IDTchecker detected the changes made to the interrupt handler code for `0x01` and `0x03` by identifying that the dispatcher code size was mismatched. Furthermore, IDTchecker detected that the offset of the start address of the handler from the driver base address in the infected VM1 had a value of `F7C477C0`. Since the maximum address range for the kernel functions was `F7C3A000`, the address detected was outside the address range of the kernel code and, furthermore, did not match the value `8053d4E4` in VM2, which was

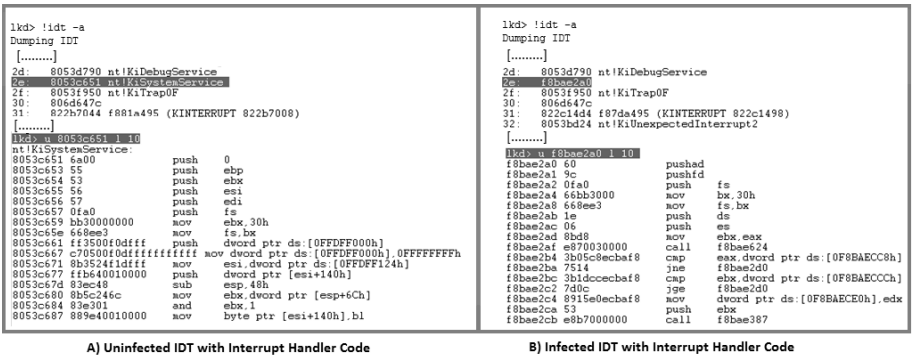


Figure 6. IDT and interrupt handler dumps.

inside the address range of the kernel code. Further examination of the assembly dump provided by the IDTchecker revealed that the expected assembly instructions were overwritten by the interrupt hooking.

- **STrace Fuzen Interrupt Hooking:** This experiment used the `STrace Fuzen` [3] malware, which hooks the IDT on the system service descriptor table (SSDT) interrupt vector `0x2E`. When an application needs the assistance of the operating system via SSDT, `NTDLL.DLL` issues interrupt `0x2E` to transfer from user space to kernel space. The malware saves the address of the original interrupt handler and changes it to the address of its own code. When an application makes a request via the SSDT, the hook is called before the kernel function in the SSDT.

  We used two identical VMs that ran Windows XP (SP2). The malware was executed on one machine and the changes were observed using the WinDbg Windows debugger; the changes were then compared with those in the uninfected machine. Figure 6 shows the IDT and interrupt handler dumps in WinDbg before and after the `STrace Fuzen` malware infection. Note that the pointer for the `0x2E` vector and the interrupt handler code in the infected machine were modified. IDTchecker successfully detected both the modifications.

5.3 Runtime Performance

This section discusses the runtime performance of IDTchecker for guest VMs that were idle and for VMs that were exhaustively using their resources. It also discusses the impact of IDTchecker on guest

VM resources along with the memory overhead of IDTchecker in the privileged VM.

5.3.1 Best Case and Worst Case Scenarios. The best and worst running times for IDTchecker were identified by tests using idle and fully-loaded VMs. For the best case scenario, the guest VMs remained idle so that IDTchecker would have all the available system resources. In the worst case scenario, the guest VMs executed resource-intensive processes that consumed most of the system resources (CPU, RAM and I/O), leaving IDTchecker very limited physical resources for execution.

Figures 7 and 8 show the execution times of IDTchecker and its components for different numbers of idle VMs and fully-loaded VMs, respectively. The figures show similar runtime patterns for IDTchecker components, with Code-Extractor consuming most of the resources. This is because, unlike Table-Extractor and Info-Extractor, Code-Extractor has to access guest VM memory several times in order to retrieve different chunks of memory corresponding to interrupt vectors in the IDT. For instance, if there are 100 interrupt gates in the IDT, then Code-Extractor has to access memory 300 times, once for each of the three associated memory elements per interrupt gate. On the other hand, Table-Extractor has to access memory only twice: once to access the IDT and the second time to access the GDT.

Linear growth is observed in the execution time of IDTchecker as the number of VMs is increased. This is because IDTchecker accesses the VMs sequentially, reading the memory of one VM at a time. This is also the reason why Code-Extractor shows the same behavior as IDTchecker. On the other hand, the execution time of Integrity-Checker remains constant as the number of VMs is increased. This is because Integrity-Checker does not access the guest VM memory, and only needs to apply the four rules to the processed VM data.

5.3.2 Impact on Guest VM Resources. Because the components of IDTchecker execute outside a guest VM, there should be a minimal performance impact on the guest VM resources. Figures 9 and 10 show the processor and memory usage for an almost idle guest VM. The boxes in the figures show zoomed-in portions from the original graphs when IDTchecker was accessing guest VM memory. The slight sign of disturbance is caused by the monitoring of system resource usage from within the VM. The boxes correspond to the time frames when IDTchecker was running on the guest VM and extracting the tables and memory chunks from the physical memory of the VM. The graphs show

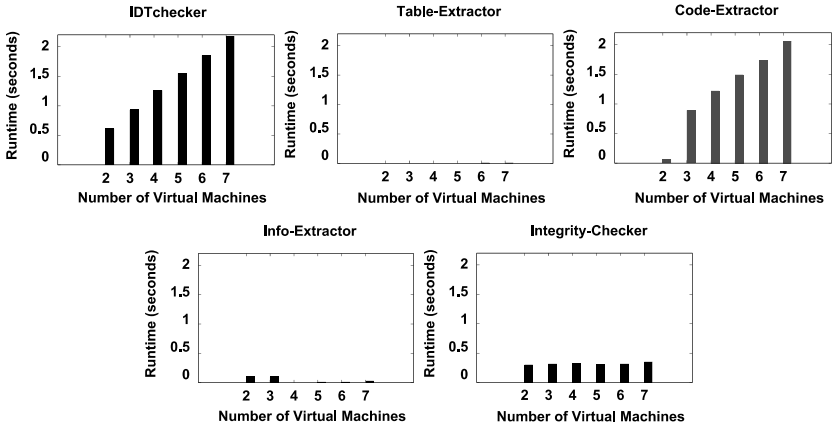


Figure 7. Execution times of IDTchecker and its components for an idle VM.

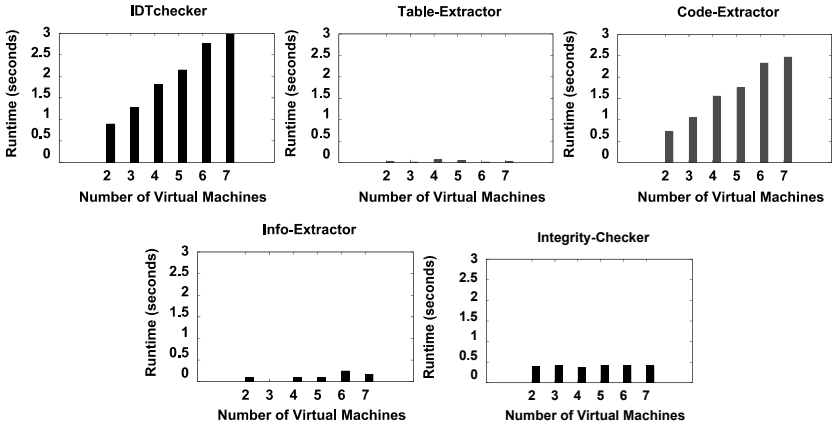


Figure 8. Execution times of IDTchecker and its components for a fully-loaded VM.

that no significant perturbations are induced by IDTchecker on the processor and memory resources of the guest VM. Thus, we can conclude that IDTchecker does not have a significant impact on guest VM resources.

5.3.3 Memory Overhead. Figure 11 shows the memory overhead of IDTchecker on a privileged VM running Fedora 16. The boxes correspond to the time frames when IDTchecker was running on the VM, and they show zoomed-in portions from the original graphs. Approximately 500 MB RAM was available to IDTchecker because the other seven guest VMs occupied 7 GB of memory and the Fedora operating system occupied 500 MB. During the tests, the machine was idle with

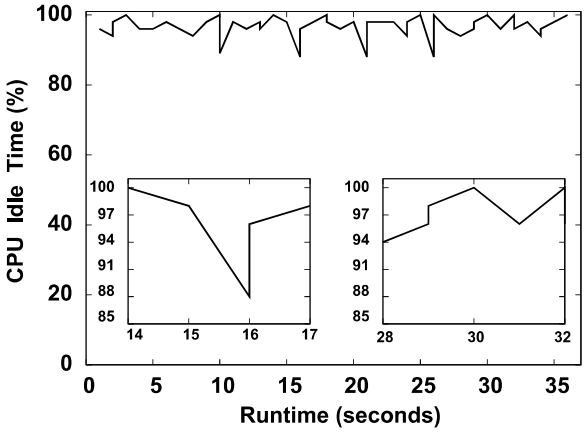


Figure 9. IDTchecker CPU usage.

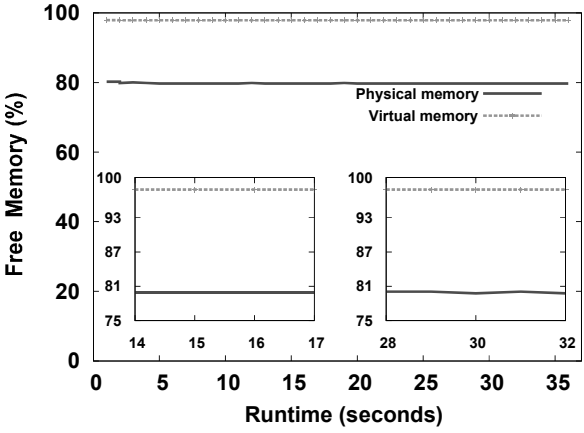


Figure 10. IDTchecker memory usage.

the exception of the usage monitoring process. A 10 to 15 MB perturbation in memory usage was caused by IDTchecker – this is just 2 to 3% of the total available memory. No usage of virtual memory was observed. The boxes in Figure 11 show the zoomed-in portions of the perturbations related to physical memory usage.

6. Conclusions

The IDTchecker tool is designed to check the integrity of IDTs and the corresponding interrupt handling code in guest VMs running in cloud environments. IDTchecker provides alerts when a VM is compromised using a majority vote based on the outputs of a set of pre-defined rules. However, IDTchecker cannot identify exactly which VM is compromised.

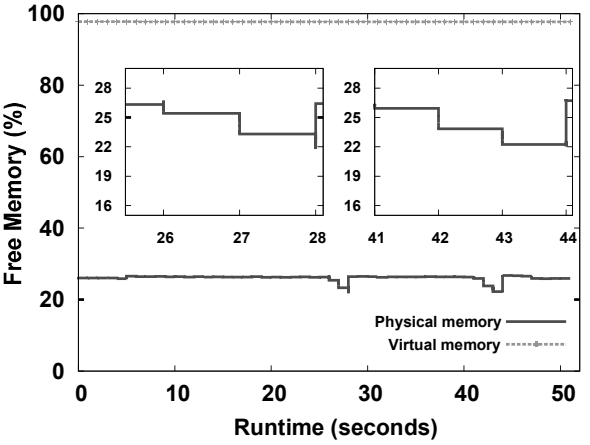


Figure 11. IDTchecker memory overhead for a privileged VM.

Thus, the tool is best used to detect the first signs of compromise, which can then trigger a resource-intensive forensic investigation to find the root cause of the problem.

Experiments demonstrate that IDTchecker is effective at detecting modifications to pointer values and interrupt handler code. Runtime performance testing of IDTchecker shows linear growth in execution time as the number of VMs is increased. Also, IDTchecker has a minimal impact on guest VM resources such as processor and memory.

Acknowledgement

This research was supported by NSF Grant No. CNS 1016807.

References

[1] I. Ahmed, A. Zoranic, S. Javaid and G. Richard III, ModChecker: Kernel module integrity checking in the cloud environment, *Proceedings of the Forty-First International Conference on Parallel Processing Workshops*, pp. 306–313, 2012.

[2] A. Bassov, Hooking the kernel directly (`www.codeproject.com/Articles/13677/Hooking-the-kernel-directly`), 2006.

[3] J. Butler and G. Hoglund, *Rootkits: Subverting the Windows Kernel*, Addison-Wesley, Boston, Massachusetts, 2005.

[4] T. Garfinkel and M. Rosenblum, A virtual machine introspection based architecture for intrusion detection, *Proceedings of the Network and Distributed System Security Symposium*, pp. 191–206, 2003.

[5] Intel, Intel 64 and IA-32 Architectures Software Developer's Manuals, Santa Clara, California (`www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html`), 2013.

[6] Kad, Handling the interrupt descriptor table for fun and profit, *Phrack*, vol. 0x0b(0x3b), 2002.

[7] G. Kroah-Hartman, Signed kernel modules, *Linux Journal*, vol. 2004(117), article no. 4, 2004.

[8] P. Loscocco, P. Wilson, J. Pendergrass and C. McDonell, Linux kernel integrity measurement using contextual inspection, *Proceedings of the Second ACM Workshop on Scalable Trusted Computing*, pp. 21–29, 2007.

[9] Microsoft, Digital Signatures for Kernel Modules on Windows, Redmond, Washington (`msdn.microsoft.com/en-us/library/windows/hardware/gg487332.aspx`), 2007.

[10] mxatone and ivanlef0u, Stealth hooking: Another way to subvert the Windows kernel, *Phrack*, vol. 0x0c(0x41), 2008.

[11] W. Oney, *Programming the Microsoft Windows Driver Model*, Microsoft Press, Redmond, Washington, 2002.

[12] Opdis Project, Opdis (`mkfs.github.com/content/opdis`).

[13] OpenSSL Core and Development Team, OpenSSL Cryptography and SSL/TLS Toolkit (`www.openssl.org`), 2009.

[14] pragmatic, (Nearly) complete Linux loadable kernel modules: The definitive guide for hackers, virus coders and system administrators (`newdata.box.sk/raven/lkm.html`), 1999.

[15] J. Rutkowska, System virginity verifier: Defining the roadmap for malware detection in Windows systems, presented at the *Hack in the Box Conference*, 2005.

[16] sd and devik, Linux on-the-fly kernel patching without LKM, *Phrack*, vol. 0x0b(0x3a), 2001.

[17] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn and P. Khosla, Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems, *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, pp. 1–16, 2005.

[18] S. Skape, Bypassing PatchGuard on Windows x64 (`uninformed.org/?v=3&a=3&t=sumry`), 2005.

[19] M. Suiche, IDTGuard v0.1 December 2005 Build (`www.msuiche.net/2006/12/10/idtguard-v01-december-2005-build`), 2005.

[20] VMI Tools Project, LibVMI (`code.google.com/p/vmitools`).

[21] Volatility Project, The Volatility Framework (code.google.com/p/volatility).

[22] Volatility Project, Volatility Plugin (code.google.com/p/volatility/source/browse/trunk/volatility/plugins/linux/check_idt.py?spec=svn2273&r=2273).

[23] Xen Project, Xen, Cambridge, United Kingdom (www.xenproject.org).

VII

FORENSIC TOOLS

Chapter 22

COMPARISON OF THE DATA RECOVERY FUNCTION OF FORENSIC TOOLS

Joe Buchanan-Wollaston, Tim Storer and William Glisson

Abstract Commercially-available digital forensic tools are often large, expensive, complex software products, offering a range of functions to assist in the investigation of digital artifacts. Several authors have raised concerns about the reliability of evidence derived from these tools. This is of particular importance because many forensic tools are closed source and, therefore, are only subject to black box evaluation. In addition, many of the individual functions integrated into forensic tools are available as standalone products, typically at a much lower cost or even free. This paper compares – rather than individually evaluates – the data recovery function of two forensic suites and three standalone non-forensic commercial applications. Experimental results demonstrate that all the tools have comparable performance with respect to the data recovery function. However, some variation exists in the data recovered by the tools.

Keywords: Digital forensic tools, data recovery, testing

1. Introduction

Forensic tools are used by thousands of digital forensic professionals around the world. The functionality of forensic tools varies, although several features appear to be provided consistently, including hard disk image preparation and storage, data hashing of entire disk images or individual artifacts, disk image mounting and filesystem reconstruction, data presentation and visualization, and data carving of damaged images and deleted file contents.

Data related to the market shares of forensic tools appears to be a closely guarded secret. However, a review of online forums, corporate

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 331–347, 2013.

websites and the research literature gives the impression that major vendors of forensic tools for personal computers are Guidance Software and AccessData, who market the EnCase [13] and Forensic Toolkit (FTK) [1] software suites, respectively. These software suites are widely used, perhaps due to the integration of several forensic applications in a common product. In addition, the provision of "push button" graphical user interfaces reduces the level of training and computing expertise required to conduct a forensic investigation.

Both Guidance Software and AccessData provide compelling arguments for employing their software in digital forensic investigations. For example, Guidance Software describes the EnCase suite as "the industry-standard computer forensics investigation solution" [13] while AccessData claims that FTK is "the most advanced computer forensics software available" [1]. Moreover, both vendors maintain that their software products are designed and validated to meet the standards of forensic evidence. FTK is described as a "court-validated digital investigations platform" [1]. EnCase is described similarly as having "an unsurpassed record of court acceptance" [13]. These claims are difficult to assess. In particular, it is unclear what level of scrutiny has been applied to the evidence produced by the forensic tools and what standards they have been assessed against.

A few independent reviews have compared the functionality and performance of forensic software suites. *SC Magazine* conducts annual group tests of forensic software, typically considering around eight to ten products [22]. Separately, the National Institute of Standards and Technology (NIST) has implemented the Computer Forensics Tools Testing (CFTT) Program [20]. This program has developed a draft validation framework for forensic data recovery tools [19]. To date, the framework has only been applied to a small selection of forensic software suites [15].

Anecdotal evidence suggests that forensic software suites may not be defect-free and that the results from different toolkits are likely to differ to some degree. The February 2008 release of FTK version 2 received bad press [2, 9, 23]. One review described the software as "an unmitigated disaster" [23]. It noted that users reported problems while installing and running the software. The documented minimum computer specification was also reported to be inadequate for the software. Mercuri [18] has noted that CFTT Program tests of EnCase and FTK revealed defects in the hard disk image preparation processes. Both tools were unable to recover some data from NTFS-formatted logical disk partitions.

Forensic tools can be expensive to purchase and operate. As of the middle of 2011, a single user license for EnCase or FTK was approximately $2,995 and the annual cost of software maintenance and support

was an additional $599 for EnCase and $840 for FTK. The hardware requirements for these tools also impose significant costs. For example, the recommended system specification for FTK (version 3) includes an Intel i9 Dual Quad Core Xeon Processor, 12 GB RAM and a 160 GB solid state hard drive dedicated entirely to an Oracle database for case management. The online shopping service provided by a popular UK vendor was used in November 2011 to prepare an estimate for a machine with the minimum required specification. The estimate suggests that the recommended platform would cost approximately £2,600 (about $4,100), not including peripheral equipment and sales taxes.

Despite the popularity of commercial forensic suites, numerous applications are available that provide equivalent functions at a much lower cost or even at no cost. Assembling a toolkit of mass market applications that has equivalent functionality to a forensic suite is an attractive proposition, not just to reduce costs. The provision of a supplementary, low-cost toolkit can ease the process of validating results generated by forensic suites and increase confidence in the reliability of evidence.

This paper compares the results of data recovery by digital forensic toolkits and mass market applications. Data recovery is the extraction and presentation of file contents from a disk image formatted using a known filesystem. This definition excludes the recording of the disk image itself and the recovery of file contents stored in areas of the disk image that are not managed by the filesystem.

The comparative approach mimics the situation faced by a digital forensic practitioner when confronted with a previously unseen data storage device. If the device is processed using different data recovery applications, there is the potential for the results to differ. This contrasts with the validation method adopted by NIST [19], in which a ground truth known data set of files or other data items is prepared and validated. In the work presented here, no assessment is made of the correctness of the forensic tools. Rather, the intention is to provide a means for quantifying the extent to which the data recovery results for different tools differ.

The comparative experiments described in this paper focused on several disk images representing the evolution of data stored on a computer as a result of user actions. The disk images were processed using a selection of recovery tools and the comparison results are presented. A holistic data comparison of the tools is conducted and the recovery of known marker files that were deliberately added to the disk image are assessed. Several experimental findings and their relation to previous literature are discussed, along with conclusions related to forensic investigations and tool validation.

2. Experimental Method

The purpose of this research is to compare the data recovery capabilities of a selection of software tools on a typical desktop personal computer setup. The personal computer is assumed to have office applications, web browsing, email communications and media playback. Some progress has been made in identifying realistic data sets for forensic tool analysis based on data found in re-sold hardware [10, 12, 16]. However, we are not aware of any research that establishes a characteristic data set for tool testing. We chose to develop our own data set to maintain control of the experiments.

A Windows XP operating system and a selection of desktop applications were installed on a pristine hard disk. A number of marker files representing what might be found in a typical user system were copied to the disk or created directly using the installed applications. Selected files were then deleted from the hard disk via the operating system user interface. Finally, a number of additional files were added to the disk, potentially overwriting files deleted by the user.

An image of the disk was taken at each stage of the experiment using FTK Imager; these images are referred to as Image1 through Image6. To gain assurance about the correctness of the images produced by FTK Imager (version 2.9.0.1385), the imaging process was repeated for the final image using the `dcfldd` tool [14]. MD5 hashes of the `dcfldd` and FTK images were computed using the `dcfldd` and FTK tools. All four image hashes for Image6 matched. The files were then recovered from the images using several data recovery tools. All the files were hashed and the file hashes and reported filesystem paths were analyzed for variations.

2.1 Target System Setup

A 20 GB hard disk was chosen for the installation as it was deemed large enough to hold the operating system along with a variety of files (including photos and videos) while being relatively quick to image and process. Typical hard disks available in a new computer (as of 2012) range from 500 GB for a laptop to 2 TB for a desktop, but using such large disks would have considerably increased the time required for imaging and processing the disk multiple times.

Windows XP Professional SP3 was installed on the disk with a single user account. The operating system installation process formatted the target hard disk using NTFS. Software for a Netgear Wireless USB Adapter, Internet Explorer 8, Firefox 5, Microsoft Office 2007 and Skype 5.5 were also installed. At this point, the disk was imaged to create Image3. Windows was then activated. Earlier images (Image1 and Image2)

were recorded, but these images are not relevant to this work and are, therefore, not discussed.

2.2 Image Preparation with User Marker Files

Internet Explorer and Firefox were opened and a number of websites were visited using each browser. In each case, the URL was typed directly into the browser address bar instead of using a search engine or link to access the website. The sites visited were selected because they were known to be static sites without changing content such as advertising banners and graphics. The advantage of this approach is that if further analysis of the browser caches were to be performed, the origin of each web page and image file could be identified easily.

Skype was then opened and a voice call was initiated to one contact, followed by a short instant message session where a message was sent to the contact and a response received from the same contact. This created a history file for Skype that could be analyzed further, if required. Outlook was then opened, an email account was configured and a test email was sent to a contact. A reply was received back from the same contact. Four appointments were then added to the calendar. This resulted in user content being stored in the `outlook.pst` data file.

The next step was to open Windows Explorer and create a new folder in the My Documents folder. A new blank text document was created in the new folder, some text was added and the file was saved. A number of files were prepared and saved on an external hard drive. The files corresponded to those found on a typical computer, including word processing documents, spreadsheets, PDF documents, photographic images, plain text files, audio, video and executable files. The external hard drive was connected to the computer via a USB cable and the files were copied across to folders in the My Documents folder. A new Microsoft Word document was then created, some text was added and the file was saved. A new Microsoft Excel spreadsheet was also created. A total of 86 user files were added to the My Documents folder. At this stage, the disk was imaged to create Image4.

Table 1 summarizes the marker files, including the file IDs, file types and file extensions. The table also shows:

- **Manipulations to Images4 through Image6:** a = Added before Image4; b = Left in Recycle Bin for Image5; c = Deleted and removed from Recycle Bin; d = Permanently deleted in Image4; e = Altered before Image6; f = Added before Image6.
- **Recoveries from Image5:** g = EnCase; h = FTK; i = Recuva; j = R-Studio; k = Stella Phoenix.

Table 1. Summary of marker files.

File IDs	File Type	Ext	Manipulations/Recoveries a b c d e f g h i j k l m n o p
1	A - Created documents	docx	✓ ✓ ✓✓✓✓✓
2	A - Created documents	txt	✓ ✓
3	A - Created documents	xlsx	✓ ✓ ✓✓✓✓✓
4,5	Excel spreadsheets	xls	✓✓ ✓✓✓✓✓✓✓✓✓✓
6	Excel spreadsheets	xls	✓ ✓ ✓✓✓✓✓✓✓✓✓✓
7	Excel spreadsheets	xls	✓ ✓
8	Excel spreadsheets	xls	✓ ✓ ✓✓✓✓✓
9-12	Movies	avi	✓✓ ✓✓✓✓✓✓✓✓✓✓
13-16	Movies	avi	✓ ✓ ✓✓✓✓✓
17,18	Movies	iso	✓ ✓
19	Music	mp3	✓✓ ✓✓✓✓✓
20-23	Music	mp3	✓✓ ✓✓✓✓✓✓✓✓✓✓
24,25	Music	mp3	✓ ✓ ✓✓✓✓✓✓✓✓✓✓
26-28	Music	mp3	✓ ✓ ✓✓✓✓✓
29,30	Other files	exe	✓✓ ✓✓✓✓✓✓✓✓✓✓
31,32	Other files	exe	✓ ✓ ✓✓✓✓✓
33,34	Other files	psd	✓✓ ✓✓✓✓✓✓✓✓✓✓
35,36	Other files	psd	✓ ✓ ✓✓✓✓✓
37-41	PDF files	pdf	✓✓ ✓✓✓✓✓✓✓✓✓✓
42-46	PDF files	pdf	✓ ✓ ✓✓✓✓✓
47-61	Photos	jpg	✓✓ ✓✓✓✓✓✓✓✓✓✓
62-76	Photos	jpg	✓ ✓ ✓✓✓✓✓
77,78	Text files	txt	✓✓ ✓✓✓✓ ✓✓✓✓
79	Text files	txt	✓ ✓ ✓✓✓✓✓✓✓✓✓✓
80-81	Text files	txt	✓ ✓ ✓✓✓✓✓
82	Word documents	doc	✓✓ ✓✓✓✓✓✓✓✓✓✓
83	Word documents	doc	✓ ✓ ✓✓✓✓✓✓✓✓✓✓
84,85	Word documents	doc	✓ ✓ ✓✓✓✓✓
86	Word documents	docx	✓✓ ✓✓✓✓✓✓✓✓✓✓
87	A - Created documents	docx	✓ ✓✓✓✓✓
88	A - Created documents	xlsx	✓ ✓✓✓✓✓
89	Excel spreadsheets	xls	✓ ✓✓✓✓✓
90-93	Excel spreadsheets	xlsx	✓ ✓✓✓✓✓
94-96	Movies	iso	✓ ✓✓✓✓✓
97	Movies	iso	✓ ✓✓✓✓
98,99	Movies	mts	✓ ✓✓✓✓✓
100	Music	mp3	✓
101-109	Music	mp3	✓ ✓✓✓✓✓
110-118	Other files	exe	✓ ✓✓✓✓✓
119	Other files	msi	✓ ✓✓✓✓✓
120-129	PDF files	pdf	✓ ✓✓✓✓✓
130-159	Photos	jpg	✓ ✓✓✓✓✓
160-164	Text files	txt	✓ ✓✓✓✓✓
165	Word documents	doc	✓ ✓✓✓✓✓
166-169	Word documents	docx	✓ ✓✓✓✓✓

- **Recoveries from Image6:** l = EnCase; m = FTK; n = Recuva; o = R-Studio; p=Stella Phoenix.

A number of the files that had been copied across to the disk were deleted. Specifically, 81 files were moved to the Recycle Bin. Of these files, 42 were then removed from the Recycle Bin. Two files were reported as being permanently deleted by a Windows prompt because the files were too large for the Recycle Bin. The browsing histories were deleted from Internet Explorer and Firefox. Calendar items and all emails were deleted from Outlook, and the Deleted Items folder was then emptied. The Skype history was cleared. Approximately half of the files in the Recycle Bin were deleted, with a record being kept of the files that remained. The disk was then imaged again to create Image5.

The external hard drive was reconnected to the computer and another selection of 81 pre-prepared files were copied across to sub-folders in the My Documents folder. The copied files almost completely filled the remaining space on the disk, leaving a small amount of space for files created by the operating system such as during Internet browsing. This replicates behavior in which a user fills up a hard disk with data and is required to remove some old data to free up space. Firefox and Internet Explorer were opened and a number of websites were visited using each browser, again by typing the URLs directly into the address bars. Two new documents were created in Microsoft Word and Excel, text was added and the files were saved. Skype and Microsoft Outlook were then used and the disk was imaged again to create Image6.

2.3 Data Recovery Procedures

The following five tools were selected for comparison of data recovery functionality:

- EnCase (Guidance Software EnCase version 7.01.02.01)
- FTK (AccessData Forensic Toolkit version 3.1.2.2359)
- Recuva (Piriform Recuva version 1.40.525)
- R-Studio (R-TT R-Studio version 5.4, build 134130)
- Stellar Phoenix (Stellar Phoenix Windows Data Recovery version 4.2 Home Edition)

The first two tools are parts of commercially-available digital forensic software suites. They were selected due to their popularity with forensic practitioners and their availability for our research. The remaining three

tools are mass market data recovery applications that are not advertised as suitable for digital forensic recovery. A plethora of options exist in this category; the three tools were selected as representative of the market and available feature sets.

All five tools were installed on an HP Z400 workstation running Windows 7 Enterprise 64-bit with a Intel Xeon Dual Core W3503 Processor (2.40 GHz) with 8 GB RAM and a 750 GB hard disk. The disk images were all stored on this workstation.

Each of the applications presents a different selection of options to the user for the purpose of configuring the recovery process. This variability in features and presentation makes the direct comparison of the tools rather challenging. The configuration of each tool is presented here to support experimental repeatability. All the options cannot be described exhaustively, and the narrative records where non-default options (as presented to the user) were selected for an application.

For EnCase and FTK, the first step in processing evidence is to start a new case. A case contains all the evidence, bookmarks, information and reports, and allows searches of the evidence. The three mass market data recovery tools have no case management options. In the case of the EnCase and FTK forensic suites, the number of options that may be selected before the scanning and recovery processes is much greater than for a tool that only recovers data.

EnCase, FTK and R-Studio are designed to allow a raw disk image to be loaded directly into the software. The other two tools, Recuva and Stellar Phoenix, do not offer this capability. For these tools, the image must first be mounted in Windows as a logical drive; this was achieved using Mount Image Pro version 4.48(828) [11].

The five tools were configured as follows:

- **EnCase:** A new case was created and the image was added as evidence to this case with default options. The "Recover Folders" task was selected (only) and the processing was started.
- **FTK:** A new case was created and the image was added as evidence. All the options were disabled except for the generation of MD5 hashes and the processing was started.
- **Recuva:** The mounted drive was selected. In the recovery dialog, the options selected were "Show files found in hidden system directories," "Show zero byte files," "Deep scan" and "Scan for non-deleted files."
- **R-Studio:** The disk image was opened and the "Whole disk scan" and "Detailed view during scan" options were selected.

- **Stellar Phoenix:** The "Search Drive" dialog was opened for the image and the "Physical Drive" method was used. The options "Deep Scan" and "Advanced Scan" were selected.

3. Experimental Results

The results reported in this section pertain to the data recovered from Image4, Image5 and Image6. These images represent the state of the target hard disk after user activity was simulated.

3.1 Analysis of Recovered Files

All the tools recovered between 13,500 and 15,200 files from each of the three images, Image4, Image5 and Image6. The analysis below does not assume that any one tool provides an accurate baseline for the number of files to be recovered. Consequently, it is not possible to report the absolute proportion of files recovered by any one tool. Instead, the differences between the tools are investigated. Several reasons for the variations between tools were identified; these are discussed below.

The forensic suites, EnCase and FTK, recover space that is not allocated by the filesystem as multiple logical files. EnCase provides options for the user to specify the size of each file created from unallocated space while FTK automatically decides how to divide the unallocated space into files and names the files according to the cluster number.

File slack is the disk space between the end of the file content and the end of the last cluster in which is the file is saved. FTK recovers file slack as files named:

```
<path>\<filename>.<extension>.FileSlack
```

FTK exported 1,244 slack space files from Image4, 1,171 from Image5 and 1,200 from Image6. EnCase exports slack space files for every item in the case, even when a file contains no data.

The data recovered from unallocated space and file slack is of interest to a digital forensic practitioner. These regions of a disk may contain remnants of deleted files. However, the research presented here excludes the recovery of data not managed by the filesystem, so these results are not included in the analysis.

Many filesystems support supplementary data attributes called alternate data streams (ADSs) in addition to the default stream [5]. An ADS can be used to store supplementary information about a file such as the zone identifier of a file downloaded from a web server. An ADS can also be used to store data in a manner that is not obvious to a casual browser of a filesystem. Not all filesystems support ADSs, so different recovery

tools present this data in different ways. EnCase and FTK recover ADSs that are used to denote zone identifiers as separate files. These files are named as follows:

```
<path>\<filename>.<extension>.<Zone.Identifier>
```

R-Studio recovers ADSs and incorporates them into the original file if the host filesystem supports them. Recuva and Stella Phoenix do not recover ADSs.

Every directory in an NTFS filesystem contains `$I30`, a directory index file that lists the directory files and sub-directories [5]. FTK recovered some of these files and labeled them `$I30`. The other four tools did not recover directory index files.

FTK recovered files from the root folder into two folders: [root] and [root][1008]. After investigating this with the recovered files from Image4, we determined that no files were duplicated and that it was simply a matter of presentation. For the purposes of comparing the result across the tools, all the files were regarded as having been recovered to one root folder. It was also apparent that FTK had appended file ID numbers (e.g., 2708) to file names starting with the $ symbol. These files were removed before analyzing the results any further.

The tools differ in their presentations of filenames for files that are in the user's Recycle Bin. EnCase and Recuva show the original filename whereas FTK, R-Studio and Stellar Phoenix show renamed files such as `Dc1.xls` and `Dc5.avi`. The naming convention is as follows:

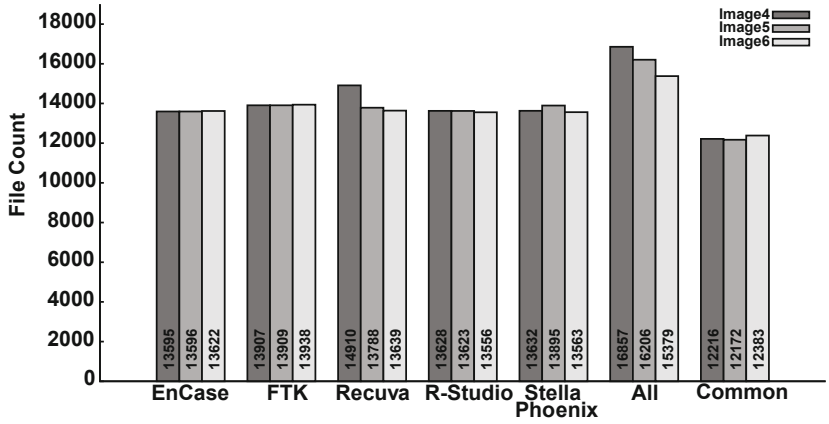
```
D<original drive letter of file><#>.<original extension>
```

The mapping of the original filename to the renamed file is found in the `INFO2` file, a normally-hidden file that is created the first time that the Recycle Bin is used [8].

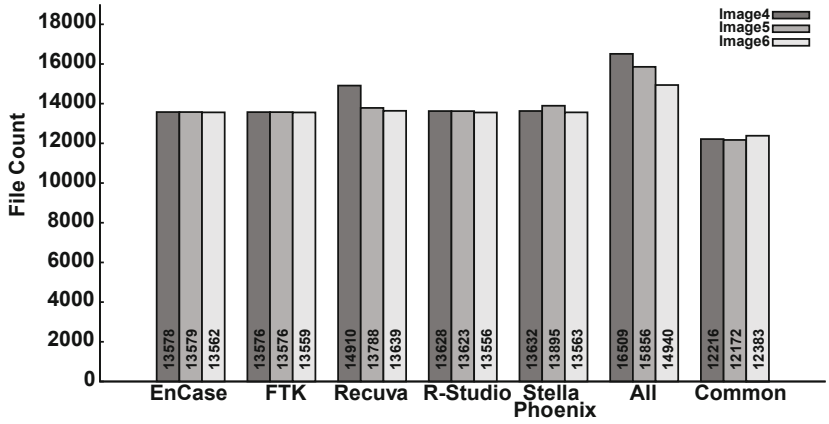
These sources of data are of potential interest to a digital forensic practitioner, so it is desirable that they are included in a data recovery process. In addition, both sources of data are managed by the NTFS filesystem. Consequently, two analyses are presented below:

- All recovered files excluding unallocated space and file slack.
- All recovered files excluding unallocated space, file slack, Zone Identifier alternate data streams and `$I30` index files.

Figure 1 shows the numbers of files recovered from Image4, Image5 and Image6 by each of the recovery tools. Figure 1(a) compares the numbers of files recovered by each tool from the disk images. The figure also shows the total number of files recovered from each image using



(a) Total files.



(b) Total files excluding index and ADS files.

Figure 1. Files recovered from Image4, Image5 and Image6 using the tools.

all the tools, and the total number of common files recovered by all the tools from each image. Two files are considered identical if they have matching file paths (taking into account the adjustments described above) and matching MD5 hash values. Figure 1(b) shows the same totals, excluding the index and ADS files.

An analysis of the results reveals that, out of the total files recovered from a single image by all the tools, a single tool recovers between:

- 80.6% (EnCase from Image4) and 90.6% (FTK from Image6) when all the files are considered.
- 82.2% (FTK from Image4) and 91.3% (Recuva from Image6) when the index and ADS files are excluded.

Although the results show that no one tool recovers all the files found by all other tools, it is unclear from this analysis if the tools recover similar file types and file locations.

3.2 Analysis of User-Created Files

During the experiment, a number of marker files were copied to the disk so that their presence among the recovered files could be analyzed for each image. Some files were deleted by moving them to the Recycle Bin, and some of the files moved to the Recycle Bin were removed from it. Additional files were copied to the disk installation so that some previously-deleted files would be overwritten. This section analyzes the findings.

A total of 86 files were created within the user's My Documents folder or were copied to it. EnCase, FTK, Recuva and R-Studio successfully recovered all 86 files with hashes that matched the originals. Stellar Phoenix successfully recovered 84 of the 86 files. The remaining two files (Files 77 and 78) were substantially recovered, but two bytes in each file had been altered, so the MD5 hashes did not match. The recovery process from Image4 was repeated to confirm this result.

Table 1 summarizes the recovery of files from Image5 and Image6. A total of 83 files were deleted during the preparation of Image5. EnCase, FTK, Recuva and R-Studio successfully recovered 82 of the 86 files from Image5 with hashes matching the originals. Two files (Files 17 and 18) were not recovered at all by any of the tools and two (Files 2 and 7) were recovered, but their MD5 hashes did not match the originals. File 2, a text file, was missing some content part of the way through the file. File 7, an Excel spreadsheet, was recovered, but was substantially corrupted (approximately 40% of the bytes had been changed). Two files were not recovered by Stellar Phoenix from Image5, these were same files (Files 77 and 78) that were not successfully recovered from Image4. However, the file contents were changed at different locations compared with Image4, so the files recovered from Image4 and Image5 had different MD5 hashes and were different from the originals.

EnCase, FTK, Recuva and R-Studio successfully recovered 125 of the 169 files from Image6 with hashes matching the originals (the same files were recovered by each of these four tools). Stellar Phoenix successfully recovered 122 of the 169 files with hashes matching the originals. As in the case of Image5, two of the additional unrecovered files were text files. The third file, an ISO formatted disk image was recovered, but it had some additional content (2,048 bytes) at the end of the file compared with the original file.

In summary, EnCase, FTK, Recuva and R-Studio performed identically when recovering marker files from all three images (Image4, Image5 and Image6). Stellar Phoenix corrupted two bytes in each of two text files (even when these files had not been deleted) and added a padding of zeroes at the end of another file that had not been deleted.

3.3 Differences in Hashes due to Image Mounts

It was observed in that, under certain circumstances, different hash values were produced by two tools for identical files. This was due to the manner in which the image containing the file was mounted as a logical drive.

Mount Image Pro was used to mount images to enable data recovery for Recuva and Stella Phoenix. The MD5summer tool version 1.2.0.5 was used to compute hashes for files recovered using these tools [21]. Fourteen files from Image4 had different hashes computed by MD5summer after mounting the image with the Mount Image Pro's "Physical and Logical" mount option, compared with those computed by FTK Imager.

The "Mount File System" option for Mount Image Pro was also tested with MD5summer. When this option was used, twelve of the fourteen files that had different hashes for the "Physical and Logical" option agreed with those computed by FTK Imager. The hashes computed for the two other files matched those computed by MD5summer using the "Physical and Logical" option. However, the hashes of numerous other files did not match the hashes previously calculated using FTK Imager.

The Image4 file was also loaded into FTK Imager, both as a raw image and as a mounted drive. The mounting used Mount Image Pro with the "Physical and Logical" mount option. In both cases FTK Imager computed the same hash values for all the recovered files.

This analysis demonstrates that care must be exercised when using tools to mount a disk image as a filesystem. The way in which a disk image is mounted can result in different hashes being computed for the files in the disk image. Our future research will investigate the reason for these differences.

4. Related Work

Several authors have argued that software tools must be validated before they can be considered suitable for forensic purposes [4, 18]. NIST has created the CFTT Program [19], which develops test sets and methods for evaluating forensic software functions such as image acquisition and hashing. Mercuri [18] has commented on the apparent defects in

image acquisition functions in the EnCase and FTK forensic tool suites, as identified by the CFTT Program.

However, data recovery is perhaps an intrinsically harder function to validate than disk image creation. This is because software tools require judgments to be made about how the recovered data files are to be presented to a user. A comparative approach overcomes some of these problems by providing estimates of the differences between data recovery applications rather than setting a ground truth as an absolute standard.

Several authors have conducted empirical comparisons of digital forensic software and of software used for digital forensic purposes. Childs and Stephens [7] have assessed three Linux forensic tools, Vinetto, Pasco and `mork.pl`. Each of these tools is designed to perform a specific task and none is intended to fulfill the needs of a digital forensic practitioner who wishes to recover and analyze all the files from a device. The tool comparison conducted by Childs and Stephens is thus limited by the specific functions provided by each tool.

The use of digital forensic tools in an academic environment is discussed by Manson, *et al.* [17]. They compare the open-source Sleuth Kit [6] with EnCase and FTK, and measure the performance of each tool against prototype images designed by the authors. As in the case of our research, the disk images used were smaller than those found in typical computer systems. The images included a mobile phone SD card (size not disclosed), and 4 GB and 15 GB hard drives. A small number of files were added to the hard drives in a Windows XP SP2 installation. Manson, *et al.* used FTK (version 1.61a) and EnCase (version 5.05C), which have been superseded several times over in the intervening years. Their research concluded that open source tools provided the same results as commercial tools, although the usability of the open source tools varied and was difficult to measure [17].

Finally, Bariki, *et al.* [3] have proposed a standard for digital evidence to be used in reports generated using digital forensic tools. They surveyed the reporting functionality of three tools, including EnCase and FTK, and note the variations in the evidentiary items included in the reports. Their research concluded that a lack of standards leads to difficulty in producing quality reports for legal proceedings.

5. Conclusions

This research has compared the data recovery capabilities of five tools under identical conditions to assess the speed with which the tools complete the data recovery process and the extent of the variations between the tools in terms of the files recovered. No two tools produced identical

results, and no tool recovered all the files in a disk image ("all" is defined at the sum total of the distinct files collectively recovered by the tools). Of course, it is also possible that some files resident on the disk image were not recovered by any tool.

One conclusion is that digital forensic practitioners need to use multiple tools to obtain a higher proportion of files from a disk image. However, the variability of recovery tools raises concerns about the correctness of the results obtained. Specifically, different subsets of files are recovered by different toolkits, and the contents of the some recovered files differ. Also, the manner in which a recorded image is accessed by a recovery application can influence the results obtained. Data recovery of user-deleted files further complicates this problem. Therefore, digital forensic practitioners should take great care when relying on files recovered by a single tool.

Comparing the data recovery results of different forensic tools presents considerable challenges. Since the configuration options and user interface features were developed independently and recovered data is presented to the software and user in different ways, establishing equivalent configurations of the various forensic tools may not be possible.

The diversity of configuration options and presentation schemes is unsurprising due to the lack of an accepted standard for data recovery methods. Further research is required to understand the implications of these variations on the evidence produced in investigations. In particular, the extent to which the discrepancies between recovery methods can influence investigations must be better understood.

References

[1] AccessData, FTK, Linden, Utah (`accessdata.com/products/computer-forensics/ftk`, 2011.

[2] C. Ball, FTK 2.0: Product review, Electronic Data Discovery Update Weblog (`commonscold.typepad.com/eddupdate/2008/05/ftk-20-product.html#more`), May 8, 2008.

[3] H. Bariki, M. Hashmi and I. Baggili, Defining a standard for reporting digital evidence items in computer forensic tools, *Proceedings of the Second International ICST Conference on Digital Forensics and Cyber Crime*, pp. 78–95, 2010.

[4] B. Carrier, Open Source Digital Forensic Tools: The Legal Argument, White Paper, @Stake, Cambridge, Massachusetts, 2002.

[5] B. Carrier, *File System Forensic Analysis*, Pearson Education, Upper Saddle River, New Jersey, 2005.

[6] B. Carrier, The Sleuth Kit (www.sleuthkit.org/sleuthkit), 2011.

[7] D. Childs and P. Stephens, An analysis of the accuracy and usefulness of Vinetto, Pasco and mork.pl, *International Journal of Electronic Security and Digital Forensics*, vol. 2(2), pp. 182–198, 2009.

[8] M. Cross, *Scene of the Cybercrime*, Syngress, Burlington, Massachusetts, 2008.

[9] Forensic Focus Blog, What happened to FTK2? (forensicfocus.blogspot.com/2008/05/what-happened-to-ftk-2.html), May 20, 2008.

[10] S. Garfinkel, P. Farrell, V. Roussev and G. Dinolt, Bringing science to digital forensics through standardized forensic corpora, *Digital Investigation*, vol. 6(S), pp. S2–S7, 2009.

[11] GetData, Mount Image Pro v4, Kogarah, Australia (mountimage.com), 2011.

[12] W. Glisson, T. Storer, G. Mayall, I. Moug and G. Grispos, Electronic retention: What does your mobile phone reveal about you? *International Journal of Information Security*, vol. 10(6), pp. 337–349, 2011.

[13] Guidance Software, EnCase Forensic, Pasadena, California (www.guidancesoftware.com/forensic.htm), 2011.

[14] N. Harbour, dcfldd version 1.3.4-1 (dcfldd.sourceforge.net), 2006.

[15] M. Hildebrandt, S. Kiltz and J. Dittmann, A common scheme for evaluation of forensic software, *Proceedings of the Sixth International Conference on IT Security Incident Management and IT Forensics*, pp. 92–106, 2011.

[16] A. Jones, G. Dardick, G. Davies, I. Sutherland and C. Valli, The 2008 analysis of information remaining on disks offered for sale on the second hand market, *Journal of International Commercial Law and Technology*, vol. 4(3), pp. 162–175, 2009.

[17] D. Manson, A. Carlin, S. Ramos, A. Gyger, M. Kaufman and J. Treichelt, Is the open way a better way? Digital forensics using open source tools, *Proceedings of the Fortieth Annual Hawaii International Conference on System Sciences*, pp. 266b, 2007.

[18] R. Mercuri, Criminal defense challenges in computer forensics, *Proceedings of the First International ICST Conference on Digital Forensics and Cyber Crime*, pp. 132–138, 2009.

[19] National Institute of Standards and Technology, Active File Identification and Deleted File Recovery Tool Specification, National Institute of Standards and Technology, Draft for Comment 1 of Version 1.1, Gaithersburg, Maryland, 2009.

[20] National Institute of Standards and Technology, Computer Forensics Tool Testing Program, Gaithersburg, Maryland (`www.cftt.nist.gov`), 2011.

[21] L. Pascoe, MD5summer (`md5summer.org`), 2011.

[22] SC Magazine, Forensic tools 2006, New York (`www.scmagazineus.com/forensic-tools-2006/grouptest/37`), July 11, 2006.

[23] Where is Your Data? Weblog, Forensics: FTK 2 (`whereismydata.wordpress.com/2009/03/01/forensics-ftk-2`), March 1, 2009.

Chapter 23

SECURITY ANALYSIS AND DECRYPTION OF FILEVAULT 2

Omar Choudary, Felix Grobert and Joachim Metz

Abstract This paper describes the first security evaluation of FileVault 2, a volume encryption mechanism that was introduced in Mac OS X 10.7 (Lion). The evaluation results include the identification of the algorithms and data structures needed to successfully read an encrypted volume. Based on the analysis, an open-source tool named `libfvde` was developed to decrypt and mount volumes encrypted with FileVault 2. The tool can be used to perform forensic investigations on FileVault 2 encrypted volumes. Additionally, the evaluation discovered that part of the user data was left unencrypted; this was subsequently fixed in the CVE-2011-3212 operating system update.

Keywords: Volume encryption, full disk encryption, FileVault 2

1. Introduction

The FileVault 2 volume encryption software was first included in Mac OS X version 10.7 (Lion). While the earlier version of FileVault (introduced in Mac OS X 10.3) only encrypts the home folder, FileVault 2 can encrypt the entire volume containing the operating system – referred to as "full disk encryption." This has two major implications. The first is that there is a new functional layer between the encrypted volume and the original filesystem (typically a version of HFS Plus). This new functional layer is actually a full volume manager, which Apple calls CoreStorage. Although the full volume manager could be used for more than volume encryption (e.g., mirroring, snapshots and online storage migration), we do not know of any other applications. Therefore, in the rest of this paper we use the term CoreStorage to refer to the combination of the encrypted volume and the functional layer that links the volume to the HFS Plus filesystem. The second implication is that

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 349–363, 2013.

the boot process is modified because the user password or some other recovery token must be retrieved before the operating system can be decrypted.

Mac OS X volume encryption is similar to other volume encryption solutions. These include PGP Whole Disk Encryption [22], TrueCrypt [23], Sophos SafeGuard [21], Credant [6], WinMagic SecureDoc [24] and Check Point FDE [4].

This paper presents a detailed analysis of the FileVault 2 full disk encryption architecture, including the key derivation mechanisms and the data structures needed for decryption. Based on the analysis, an open source cross-platform library named `libfvde` was developed to decrypt and mount CoreStorage volumes. The library and the detailed documentation of the data structures used by FileVault 2 are available online [5]. The library can be used to analyze the contents of a particular file or block in an encrypted volume without having to use Mac OS and even without gaining physical access to the Apple computer in question (e.g., by booting from a Linux live CD and connecting to the machine via the Internet).

1.1 Motivation

Our main goal was to determine how FileVault 2 operates. This task involved finding how and where the encrypted volume master key is stored, how the key can be obtained from the user password or token, what other data (metadata) is available and how is used, and finally, how disk encryption and decryption are performed.

We were motivated by two factors. First, we needed a tool for digital forensic investigations. When a computer is suspected of having malware or having been the target of malicious access, it is necessary to obtain certain files from the disk. If the computer is at a distant location, it may not be feasible or convenient to transport the computer or disk to a forensic laboratory for analysis (it is not easy to remove the disk from a MacBook Air). Even if physical access is available to the computer, the native operating system cannot be trusted to extract the necessary files because of the presence of malware. Furthermore, although it is possible to access a FileVault 2 encrypted volume using another Mac computer connected via FireWire, this is a very limiting context. Our open source library (`libfvde`) does not have any of these limitations.

Our second motivation was to have a security evaluation of the system. Since FileVault 2 could be used on sensitive corporate machines, it is necessary to verify that no serious vulnerabilities exist.

2. Background

A hard disk is generally organized in multiple sections called partitions or volumes. These volumes are often structured according to a filesystem format (e.g., NTFS, FAT or HFS). It is possible to have a single disk with three volumes, where the first volume is formatted with NTFS and contains a Windows operating system, the second volume is formatted with EXT3 and contains an installation of a Linux distribution, and the third volume is formatted with FAT and only contains data (no operating system). Interested readers are referred to [3] for details about this topic.

Volume encryption is a mechanism used to encrypt the contents of an entire volume. This is sometimes incorrectly referred to as "full disk encryption." We will use the term "volume encryption" in this paper to refer to the encryption performed by FileVault 2.

One of the main problems with volume encryption is that the operating system data is also encrypted, so there is no code left to boot the system. Therefore, the minimum code required to decrypt the operating system (or enough to initialize the filesystem and decrypt the rest of the operating system) must reside elsewhere. This is a problem that is tackled differently by the various volume encryption solutions.

Another important aspect of volume encryption is key derivation. The volume is generally encrypted using an algorithm that relies on AES or other symmetric cipher (asymmetric cryptography would have too much of an impact on read/write performance). Therefore, there must be a key that can unlock the encrypted volume. FileVault 2 uses 128-bit AES keys and has a layered architecture that allows multiple users to decrypt the same volume master key.

The next important aspect of volume encryption is the encryption operation itself. This operation must be carefully designed because there are several problems that can make a straightforward implementation insecure [11]. Also, the nature of disk encryption has special requirements with regard to speed (encryption should not introduce noticeable delays) and size (individual fixed-size sectors must be encrypted individually so they can be accessed independently).

The last important problem deals with the storage of the volume master key during the system operation and sleep modes. During the boot process, the volume master key is derived from the user password or token. After this key is derived, the operating system stores it in memory in order to read and write blocks efficiently without having to derive the key on every disk access. Halderman, *et al.* [13] have shown that an attacker with temporary access to a running system can scan the memory to retrieve the volume master key and then decrypt

Figure 1. Recovery password shown when FileVault 2 is enabled.

the disk contents. This is still a general problem, but it is possible to use proprietary methods such as on-board tamper resistant memory to mitigate these attacks.

3. FileVault 2 Architecture

This section describes the architecture and key features of FileVault 2.

3.1 Enabling FileVault 2

After FileVault 2 is enabled, a series of events take place. First, the user is presented with a 24-character recovery password (see Figure 1) that can be used to access the encrypted volume, even if the user password is lost.

Next, the filesystem in the main volume is converted from the native HFS Plus type to CoreStorage (encrypted). During this operation, the user can still use the system and the ConversionStatus field in the `EncryptedRoot.plist` file (details are provided below) contains the string "Converting." After the encryption process is complete, the string is changed to "Complete." At this time, we do not know how the operating system keeps track of the encrypted blocks during the conversion process, so our tool cannot correctly mount volumes that are in the Converting state. We are continuing to investigate this situation.

In addition to the encryption itself, a new volume called Recovery HD appears alongside the main Macintosh HD volume. This new partition contains the encrypted volume master key.

Running the command `diskutil list` on a Mac OS installation with FileVault 2 enabled yields an output similar to that shown in Figure 2. The output contains the encrypted volume (`disk0s2`), Recovery HD volume (`disk0s3`), original unmodified EFI volume (`disk0s1`) and also the unlocked (unencrypted) version of the main volume (`disk1`). There

```
> diskutil list
/dev/disk0
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *121.3 GB   disk0
   1:                        EFI                         209.7 MB   disk0s1
   2:          Apple_CoreStorage                         119.0 GB   disk0s2
   3:                 Apple_Boot Recovery HD             650.0 MB   disk0s3
   4:                  Apple_HFS                         1.4 GB     disk0s4
/dev/disk1
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:                _     Apple_HFS Macintosh HD       *118.7 GB   disk1
```

Figure 2. Output of `diskutil list` run on a Mac Book Air with FileVault 2 enabled.

is also an additional partition (`disk0s4`), which we created only for testing purposes.

The Recovery HD volume contains a series of new files, including new EFI boot code to deal with the encrypted volume. Among all the files available in the new volume, the most important for FileVault 2 operation is the `EncryptedRoot.plist.wipekey` file, which is found at: *com.apple.boot.X/System/Library/Caches/com.apple.corestorage* where X changes between R, S and P. This file contains all the information needed to extract the volume master key from the user password or recovery token.

The `EncryptedRoot.plist.wipekey` file is encrypted using AES-XTS with an all-zeros "tweak key," but the encryption key is easily available in the header (first block) of the CoreStorage volume. The header block also contains other data, including the size of the entire volume (including metadata), another UUID that is used as a key to decrypt part of the metadata, and a CRC32 checksum. The checksum uses a weak CRC calculation based on the Castagnoli (CRC-32C) polynomial. The same checksum is used to validate the values in the FileVault 2 metadata structures.

After it is decrypted, the `EncryptedRoot.plist` file includes the following important entries: recovery password PassphraseWrappedKEKStruct, user password PassphraseWrappedKEKStruct and wrapped volume master key KEKWrappedVolumeKeyStruct. If multiple users are registered on the same machine, then the `EncryptedRoot.plist` file has a separate PassphraseWrappedKEK structure for each user.

3.2 Key Derivation

The volume master key is derived from the user password, private key token or recovery password. Since the volume master key is the same but the input may be different, FileVault 2 uses an intermediary key that is decrypted under different inputs: password or recovery token. This intermediary key, which decrypts the volume master key, is called the

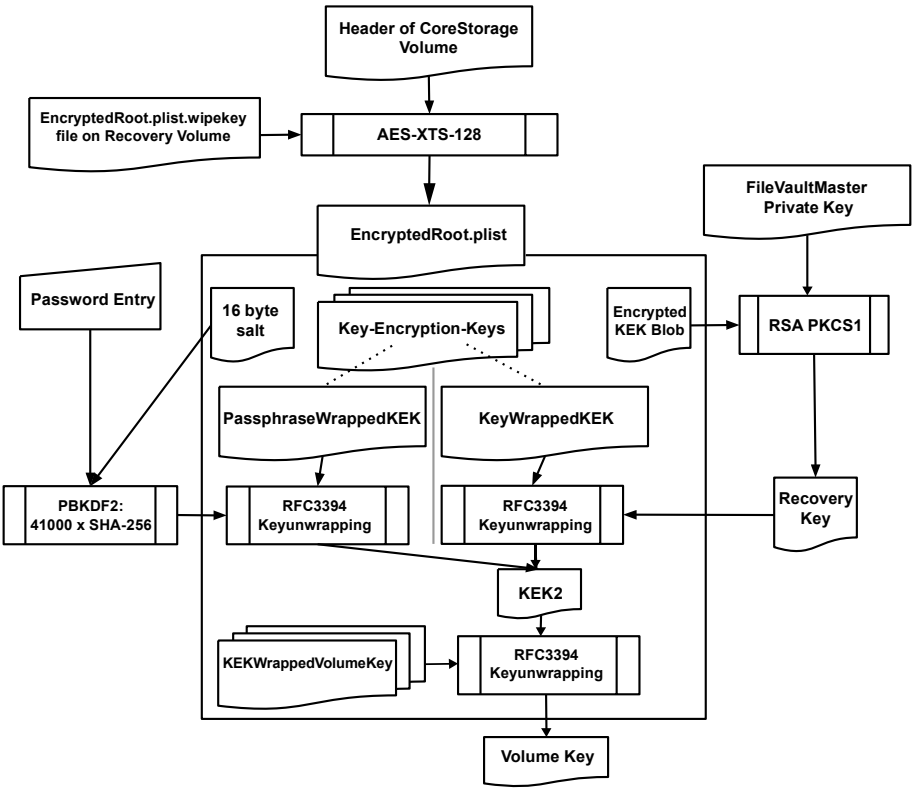


Figure 3. FileVault 2 key derivation process.

volume Key-Encryption-Key (KEK). The overall key derivation process is shown in Figure 3.

We now describe the key derivation process when the volume master key is derived from the user password. The first step is to derive the intermediary volume KEK from the given password; this is accomplished in two stages. First, PBKDF2 is used to derive a key from the user password. Then, the AES-wrapped version of the volume KEK is decrypted using this key.

The PBKDF2 Algorithm [14] derives a standard cryptographic key (of any length) from an arbitrary password string. The main objective of the algorithm is to make it difficult for an attacker to brute force all possible values of the user password. This is accomplished using two countermeasures: a salt to prevent rainbow table attacks and a large number of iterations of a pseudorandom function to increase the computational time. In the case of FileVault 2, both the salt and the number of iterations are available in the PassphraseWrappedKEK structure from

the `EncryptedRoot.plist` file. We observed that FileVault 2 uses a fixed number of 41,000 iterations of HMAC-SHA256, although we found code that allows a variable number of iterations that are multiples of 1,000 (but this code does not appear to be used).

After the PBKDF2 key is derived, it can be used to unwrap the volume KEK using the AES Wrap Algorithm [20]. The wrapped version of the volume KEK is also found in the PassphraseWrappedKEK structure. After obtaining the volume KEK, the same AES Wrap Algorithm can be applied to the data from the KEKWrappedVolumeKeyStruct structure to obtain the volume master key.

In both the unwrapping operations, the wrapped data has 24 bytes, but the unwrapped key has only sixteen bytes. This is because the first eight bytes of the unwrapped data actually contain the initial value of `0xA6` (a different value indicates a wrong decryption key), leaving only sixteen bytes for the actual unwrapped key.

The recovery password shown in Figure 1 can be used exactly as the user password. Note that the dashes are part of the password and must be included.

As an alternative decryption token – especially for organizations that need key escrow – a private key in the FileVaultMaster certificate may be used. This certificate is generally installed in */Library/Keychains/FileVaultMaster.cer*. In this case, the volume KEK is obtained from the KeyWrappedKEK structure instead of the PassphraseWrappedKEK structure. The KeyWrappedKEK structure contains a wrapped version of the volume KEK. To unwrap it, it is necessary to use a recovery key that is saved as an encrypted blob (in file `EncryptedRoot.plist`) that is protected using RSA and PKCS#1 padding. Note that the encrypted blob is added along with ExternalKeyProps to `EncryptedRoot.plist` when a certificate is used for recovery. The recovery key can be extracted from the blob using the private key from the FileVaultMaster certificate.

3.3 AES-XTS

FileVault 2 uses the AES-XTS Algorithm [16] to encrypt data. AES-XTS is a type of "tweakable encryption" that uses AES [17] as the block cipher. The XTS construction for tweakable encryption is based on XEX [19]. It uses a tweak value to encrypt each block on the volume differently, even if the plaintext is the same.

The AES-XTS encryption operation is performed per block. The blocks can be of arbitrary size, although multiples of 128 bits are generally used. For each data block to be encrypted, the algorithm expects

two keys named key_1 and key_2 (tweak key) that are 128 or 256 bits long, and a 128-bit tweak value i that is usually derived from the block offset.

AES-XTS has several advantages over alternatives such as AES-CBC: there is no requirement for an initialization vector because the tweak key can be derived from the block number; each block is encrypted differently based on the tweak value; furthermore, unlike AES-CBC, AES-XTS prevents an attacker from changing one specific bit in a data unit by XOR-ing each AES input with a different shifted version of the encrypted tweak.

FileVault 2 uses AES-XTS in several places, always with 128-bit keys. It is used to encrypt the `EncryptedRoot.plist` file, where key_1 is available on the main volume header and key_2 is 128 bits of zeroes (i.e., 16 zero bytes); the tweak value is zero and the entire file is treated as one large block. AES-XTS is also used to encrypt part of the metadata, where key_1 is the same key used to encrypt the `EncryptedRoot.plist` file, but key_2 is another value known as Physical Volume UUID, also found on the volume header. Finally, AES-XTS is used to encrypt the main volume.

In the previous section, we discussed how to derive the volume master key, which is used as key_1 with AES-XTS. However, key_2 (tweak key) is also required. Finding this tweak key was one of our most difficult tasks. Eventually, we discovered that the tweak key is derived from the volume master key and another value, Logical Volume Family UUID, which is found in the encrypted metadata.

AES-XTS encryption might be used in other places as well. We found unknown key values when performing live debugging for unknown data. We have indications that these keys are used to encrypt paged data and other memory contents.

3.4 Metadata Structures

The structures needed to decrypt the main volume, along with a header and other additional information, are stored in the CoreStorage volume listed in Figure 2. The layout of these structures within the volume is shown in Figure 4.

Two essential metadata structures are needed to decrypt the main volume: the Disk Label metadata and the encrypted metadata. The Disk Label metadata block offset and size can be found in the CoreStorage volume header. The Disk Label metadata block contains a pointer to other metadata blocks that are encrypted, so we refer to these blocks as encrypted metadata. The contents can be decrypted using AES-XTS with key_1 and key_2 from the CoreStorage volume header, starting with

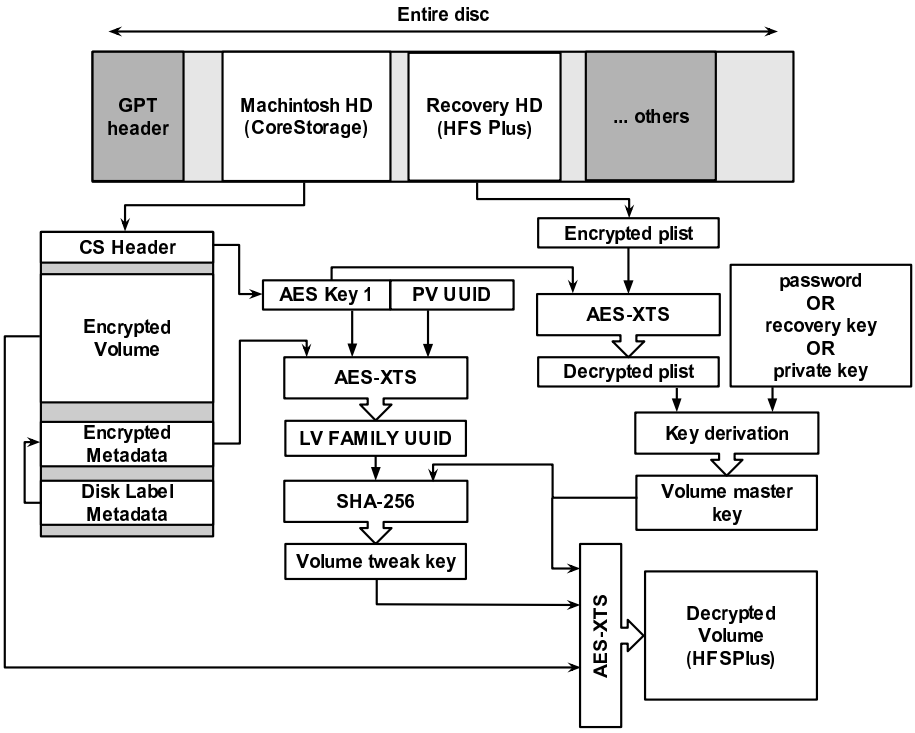


Figure 4. FileVault 2 architecture.

a tweak value of zero and using a block size of 8,192-bytes (size of an encrypted metadata block).

We are still investigating the exact structure of the metadata, which is likely related to the structure of CoreStorage. Readers who are interested in the latest information about these structures and other aspects of our work are referred to the project website [5].

The encrypted metadata contains, among other things, an XML structure with a Logical Volume Family UUID (lv.familyUUID). This UUID can be used to derive the volume tweak key by applying SHA-256 to the concatenation of the volume master key and the UUID, and retaining the first 16 bytes of the result:

$$key_2 = \mathrm{MSB}_{16}(\mathrm{SHA}_{256}(\text{volume master key} \mid \text{lv.familyUUID}))$$

3.5 Full Disk Encryption and Decryption

Having presented the building blocks of FileVault 2, we can describe the entire volume decryption process, which is illustrated in Figure 4. First, the `EncryptedRoot.plist` file is decrypted using the key from

the volume header. Then, the user password or recovery token is used to extract the volume master key. Following this, the volume tweak key is derived from the encrypted metadata and the volume master key. At this point, AES-XTS is used with the volume master key as key_1 and the volume tweak key as key_2 to decrypt the main volume, with a tweak value starting from zero and a block size of 512 bytes.

4. Security Analysis

This section presents the results of our security analysis of FileVault 2.

4.1 Recovery Password

When activating FileVault 2, the System Preferences application displays a randomly-generated 120-bit password (base32 encoded) to the end user and advises that the password should be stored securely for data recovery (see Figure 1).

The recovery password is read from `/dev/random` (through `libcsfde` and `SecCreateRecoveryPassword()` in `Security.framework`). Therefore, the security of the FileVault 2 system can be reduced to the security of the pseudorandom number generator (PRNG) used in Mac OS X Lion for `/dev/random`. Mac OS relies on Counterpane's implementation of the Yarrow PRNG [15] with modifications by Apple available as open source [1]. The Yarrow PRNG design has been rendered obsolete by Fortuna [10], which was written by the original authors.

We evaluated the seeding of the PRNG to evaluate the strength of Apple's implementation of Yarrow. Because the state of the PRNG is kept between reboots, we assume a scenario in which an end user activates FileVault 2 right after the first boot after the operating system is installed. This is the worst-case scenario where the PRNG is seeded with the least amount of entropy. During boot-time, the PRNG is seeded with 8, 20 and 332 bytes; after boot-time, the PRNG is periodically seeded with 332 bytes every 10 minutes. An attacker who guesses the seed correctly could recreate the PRNG state and predict its output, thereby determining the recovery password. The sources for the seeding are as follows:

- **Boot Seed (8 Bytes):** This seed is deterministic because it is the value of the current `microtime()` during boot.
- **Boot Seed (20 Bytes):** This seed is read from the `SystemEntropyCache` file, which contains the previous state of the PRNG before reboot. The file is written by EntropyManager every six hours and during shutdown. It contains a 20-byte output from `/dev/random`.

This seed is deterministic in our scenario because the system is booted for the first time.

- **Boot and Periodic Seed (328 Bytes):** This seed is triggered by `securityd` and the contents of the seed are collected in the kernel function `kdbg_getentropy()`. It corresponds to the core seed for the PRNG. The data contains 41 samples of `mach_absolute_time()` that returns an 8-byte nano-precision time offset for different kernel threads. We sampled the entropy seed of the PRNG over 1,000 reboots. Our estimate is that the total seeding entropy is 40 samples of eight bits of `mach_absolute_time`. This would result in 320 bits of total entropy because the nano-precision timestamps are only unpredictable in the lower bits and the higher bytes have repeating patterns. Thus, this represents a suboptimal search space, i.e., not every input to the seed is unpredictable and the amount of entropy input is less than what other operating systems seed [8, 12]. However, the search space is large enough to prevent it from being brute forced.

In a security-critical scenario, the PRNG should be reseeded by manually writing entropy to `/dev/random` before activating FileVault 2.

4.2 Plaintext Bits in Encrypted Volume

Having discovered most of the details about the operation of FileVault 2, we computed the entropy of each 512-byte block of the CoreStorage volume to verify our assumptions and also to ensure that we did not miss any data.

Figure 5 shows a bitmap of the volume encrypted with Mac OS X 10.7.2. Each pixel corresponds to a 512-byte block. Blue (dark) regions correspond to plaintext (low entropy), white regions correspond to zero or constant data such as all bytes with `0x00` or `0xFF` (zero entropy), and red (bright) regions correspond to encrypted data (very high entropy).

The bitmap shows a large block of zero data (with the exception of the first header block) at the beginning of the disk. This is followed by a large amount of encrypted data corresponding to the encrypted volume. At the end, there is a mix of plaintext, encrypted and zero data corresponding to the metadata, encrypted metadata and related structures, and the backup header (last block).

Upon examining the encrypted data portion, we discovered that there is a significant portion of plaintext (around 250 MB) in the middle of the encrypted volume. This plaintext blob contains code, dictionaries, journal metadata, error messages, debug messages and some user data.

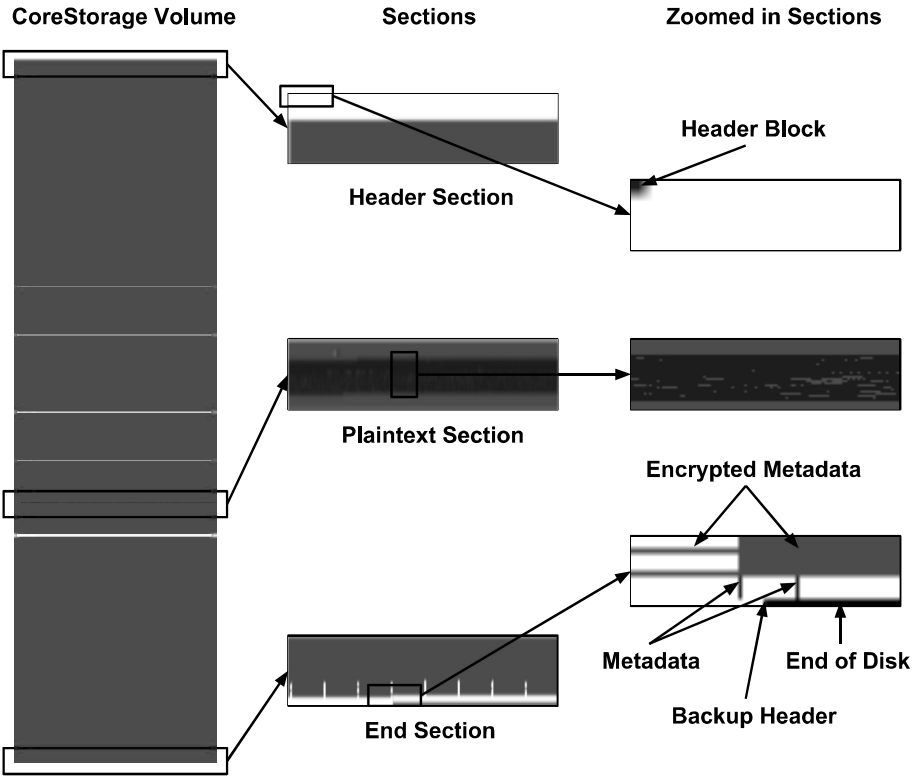


Figure 5. Entropy bitmap of the CoreStorage volume created by Mac OS X 10.7.2.

Our best guess is that this data is from the base operating system installation that was encrypted elsewhere, but that had not been wiped from the disk. Furthermore, we determined that long-used clear volumes could contain personal (possibly sensitive) data after the activation of FileVault 2. We advised Apple about this problem on February 9, 2012 (ticket ID 191364581), and it was fixed in the next update.

4.3 Possible User Password Attacks

PBKDF2 is used to slow down brute-force attacks on user passwords. Raeburn [18] reports that, for N iterations and a known salt, a 2 GHz machine can perform approximately $2^{17}/N$ PBKDF2 iterations per second. Therefore, the 2 GHz machine would require about 34 years to brute force a FileVault 2 password using a data set of 2^{32} words. However, if the password is a weak six-character common word, it could be determined within five hours. This should be taken into considera-

tion before assuming that user data is completely secure simply because FileVault 2 is enabled.

The details provided in this paper make it possible to verify that FileVault 2 users have secure passwords without requiring them to disclose their passwords. A systems administrator could use a tool that brute forces user passwords using a data set. If a password is revealed, then the administrator could request the user to choose a better password and perform the check again. This would ensure that users do not employ known weak passwords with FileVault 2.

4.4 Extracting Keys from Memory

Halderman, *et al.* [13] have shown that is possible to extract encryption keys from memory under many circumstances. FileVault 2 is also vulnerable to this attack: we could retrieve all the necessary keys from memory using the standard GNU debugger (`gdb`).

Compared with Bitlocker [9] in the TPM mode, FileVault 2 is more resistant to key extraction attacks when the computer is turned off. This is because the volume master key is never loaded in memory unless the user provides the correct authentication token. On the other hand, Bitlocker loads the volume master key from the TPM without needing the user password. Note that Bitlocker can also be used with a recovery key instead of the TPM, but this is not very common.

Dornseif [7] has shown that keys can be extracted from memory using FireWire in the DMA mode. This enables an attacker with physical access to a running system to extract the memory contents, bypassing the operating system and CPU because the transfer takes place via DMA. Fortunately, Apple addressed this problem in the OS X 10.7.2 update [2].

5. Conclusions

This paper has presented the first detailed analysis of the FileVault 2 volume encryption system that was introduced in Mac OS X 10.7 (Lion), including the key derivation mechanisms and the data structures needed for decryption. As a result of the analysis, an open source cross-platform library named `libfvde` was developed to decrypt and mount FileVault 2 encrypted volumes when the user password or recovery token are available. The library and the detailed documentation of the data structures used by FileVault 2 are available at the project website [5]. Two major points revealed by the security analysis are that the entropy of the recovery password can be increased and that a portion of user data is available as plaintext.

Acknowledgements

We thank Darren Bilby for his support of this work. We also thank Germano Caronni, Michael Cohen, Jan Monsch and Frank Stajano for their comments. This research was supported by a Google Europe Fellowship in Mobile Security awarded to Omar Choudary.

References

[1] Apple, Source Browser, Cupertino, California (`opensource.apple.com/source/xnu/xnu-1699.24.8/bsd/dev/random`), 2010.

[2] Apple, About the security content of OS X Lion v10.7.2 and security update 2011-006, Cupertino, California (`support.apple.com/kb/HT5002`), 2011.

[3] B. Carrier, *File System Forensic Analysis*, Pearson Education, Upper Saddle River, New Jersey, 2005.

[4] Check Point Software Technologies, Check Point Full Disk Encryption, San Carlos, California (`www.checkpoint.com/products/full-disk-encryption`), 2013.

[5] O. Choudary and J. Metz, `libfvde`: Library and tools to access FileVault Drive Encryption (FVDE) encrypted volumes (`code.google.com/p/libfvde`), 2013.

[6] Dell, Credant Enterprise Edition for Mac, Round Rock, Texas (`www.credant.com/ products/cmg-enterprise-edition/cmg-enterprise-edition-for-mac.html`), 2013.

[7] M. Dornseif, Owned by an iPod, presented at the *PacSec Conference*, 2004.

[8] L. Dorrendorf, Z. Gutterman and B. Pinkas, Cryptanalysis of the random number generator of the Windows operating system, *ACM Transactions on Information and System Security*, vol. 13(1), article no. 10, 2009.

[9] N. Ferguson, AES-CBC + Elephant Difusser: A Disk Encryption Algorithm for Windows Vista, Technical Report, Microsoft, Redmond, Washington, 2006.

[10] N. Ferguson and B. Schneier, *Practical Cryptography*, Wiley, Indianapolis, Indiana, 2003.

[11] C. Fruhwirth, New Methods in Hard Disk Encryption, Theory and Logic Group, Institute for Computer Languages, Vienna University of Technology, Vienna, Austria (`clemens.endorphin.org/nmihde/nmihde-A4-ds.pdf`), 2005.

[12] Z. Gutterman, B. Pinkas and T. Reinman, Analysis of the Linux random number generator, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 371–385, 2006.

[13] J. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino, A. Feldman, J. Appelbaum and E. Felten, Lest we remember: Cold boot attacks on encryption keys, *Communications of the ACM*, vol. 52(5), pp. 91–98, 2009.

[14] B. Kalisky, PKCS #5: Password-Based Cryptography Specification Version 2.0, RFC 2898, 2000.

[15] J. Kelsey, B. Schneier and N. Ferguson, Yarrow-160: Notes on the design and analysis of the Yarrow cryptographic pseudorandom number generator, *Proceedings of the Sixth International Workshop on Selected Areas in Cryptography*, pp. 13–33, 2000.

[16] L. Martin, XTS: A mode of AES for encrypting hard disks, *IEEE Security and Privacy*, vol. 8(3), pp. 68–69, 2010.

[17] National Institute of Standards and Technology, Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, Gaithersburg, Maryland, 2001.

[18] K. Raeburn, Advanced Encryption Standard (AES) Encryption for Kerberos 5, RFC 3962, 2005.

[19] P. Rogaway, Efficient instantiations of tweakable block ciphers and refinements to modes OCB and PMAC, *Proceedings of the Tenth International Conference on the Theory and Application of Cryptology and Information Security*, pp. 16–31, 2004.

[20] J. Schaad and R. Housley, Advanced Encryption Standard (AES) Key Wrap Algorithm, RFC 3394, 2002.

[21] Sophos, SafeGuard Enterprise, Abingdon, United Kingdom (`www.sophos.com/en-us/products/encryption/safeguard-enterprise.aspx`), 2013.

[22] Symantec, Symantec Drive Encryption, Mountain View, California (`www.symantec.com/drive-encryption`), 2013.

[23] TrueCrypt Foundation, TrueCrypt (`www.truecrypt.org`), 2012.

[24] WinMagic, SecureDoc for Mac, Mississauga, Canada (`www.winmagic.com/products/full-disk-encryption-for-mac`), 2013.

VIII

ADVANCED FORENSIC TECHNIQUES

Chapter 24

DETECTING COUNTERFEIT CURRENCY AND IDENTIFYING ITS SOURCE

Ankit Sarkar, Robin Verma and Gaurav Gupta

Abstract Counterfeit currency varies from low quality color scanner/printer-based notes to high quality counterfeits whose production is sponsored by hostile states. Due to their harmful effect on the economy, detecting counterfeit currency notes is a task of national importance. However, automated approaches for counterfeit currency detection are effective only for low quality counterfeits; manual examination is required to detect high quality counterfeits. Furthermore, no automatic method exists for the more complex – and important – problem of identifying the source of counterfeit notes. This paper describes an efficient automatic framework for detecting counterfeit currency notes. Also, it presents a classification framework for linking genuine notes to their source printing presses. Experimental results demonstrate that the detection and classification frameworks have a high degree of accuracy. Moreover, the approach can be used to link high quality fake Indian currency notes to their unauthorized sources.

Keywords: Counterfeit currency, detection, Indian rupees

1. Introduction

To counterfeit means to illegally make an imitation of something with the intent to take advantage of the superior value of the imitated product. Counterfeit currency refers to currency that closely resembles the original currency of a country but that is produced without the legal sanction of the government. Counterfeit currency is harmful to a nation. Fake notes increase money circulation, potentially leading to inflation. Also, the overall confidence in the currency decreases. Furthermore, individuals who innocently acquire counterfeit currency are victimized –

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 367–384, 2013.

there are usually no government policies to reimburse them for counterfeit notes that are seized. On the other hand, the individuals who produce counterfeit currency can make significant profits and finance an array of activities, many of which might be against the national interest.

Counterfeits are created in a variety of ways. The easiest and most common way is to use a high resolution scanner to capture both sides of a genuine currency note. The scanned images are then printed using a color inkjet or laser printer. This method works well for small denomination notes that are usually not scrutinized. However, it is easy to identify such counterfeit notes because of the quality of the paper that is used.

A more sophisticated method starts with a low denomination note, bleaches or washes out the ink, and then prints a higher value note. However, the production of a high quality counterfeit requires an entity to use the same raw materials and printing process that are used to produce genuine currency. This is out of the reach of individual counterfeiters due to the high cost and difficulty in procuring the raw materials and equipment. Therefore, the production of high quality counterfeits is generally state sponsored – the goal is usually to undermine the economy of the targeted state. A notable example is the "SuperDollar," a high quality imitation of the U.S. dollar. Another example is the fake Indian rupee notes produced by hostile states that are often indistinguishable from the originals.

Security features are often embedded in currency notes to identify genuine notes. Common security features include watermarks, security threads, latent images, micro-lettering, intaglio (raised print), optically variable ink and fluorescence. In addition to helping verify that a currency note is genuine, the security features deter counterfeiting. Replicating the security features increases the cost of counterfeiting, making it less profitable.

This paper focuses on high quality fake Indian counterfeit notes. While the successful identification of such notes usually requires expert examination, it is a relatively simple problem because only the integrity of the security features has to be investigated. A more difficult task is to identify the source of a counterfeit note. This requires detailed investigation by forensic scientists using expensive instruments. Also, no automatic method exists to link a counterfeit note to its source press. This paper applies digital forensic methods to address the problem, with the ultimate goal of developing an automated and scalable process that can link currency notes to their source presses.

Indian currency notes are printed at dedicated government printing presses. The number of presses is limited and great care is taken to

ensure that all the notes produced by the presses are of the same quality. However, it is extremely difficult to produce currency notes that are exactly similar. Differences creep in due to plate defects, plate wear, ink quantity and inaccuracies in the cutting process. Locard's law of exchange states that every contact leaves a trace. According to this law, all currency notes from a particular press should contain some traces of the defects. Since the plates, printing machinery and raw materials are generally not transported from one press to another, it can be assumed that these characteristics are unique to each press. Thus, notes can be grouped into classes depending on their source printing press. Consequently, this paper focuses on identifying features (with intra-class similarity and inter-class differences) that can be used to classify currency notes. The goal is that, given a genuine Indian currency note, it should be possible to classify it as having been printed at one of the government presses. The approach can then be extended to link a fake Indian currency note to its source printing press in a hostile state.

2. Related Work

This section discusses previous work related to detecting counterfeit documents and linking documents to their source devices.

2.1 Detecting Counterfeit Documents

The problem of detecting a forged document has been the subject of much research. However, most approaches involve manual methods. In the case of counterfeit currency, the approach involves visually verifying security features such as security threads, watermarks, optically variable inks and intaglio. Non-visual methods include the use of chemical analysis to verify paper quality. While manual methods are reliable, they are not scalable – most real-world scenarios require the rapid processing of currency notes as soon as they are proffered by their owners.

Few automated methods have been proposed for detecting fraudulent documents. One of the earliest approaches was proposed by Gupta, *et al.* [3], who focused on fake documents produced using scanners and printers. In particular, they discovered that a fake document has more unique colors than its genuine counterpart. Gupta, *et al.* [4] subsequently introduced two measures for detecting fake documents, variation in intensity and grey level co-occurrence matrix uniformity.

Ryu, *et al.* [9] applied machine learning in an automatic framework for detecting fake documents. They modeled detection as a classification problem with two classes, genuine and fake. An SVM classifier was used with seventeen image quality measures as features. The SVM clas-

sifier worked relatively well for forgeries using scanners and printers, but there is still scope for improvement, especially with regard to detecting counterfeit currency notes (see, e.g., [1, 7, 8, 11, 12]).

Another approach is to use intrinsic features to compute a signature for a genuine document [10]. Given a set of pre-processed (properly rotated and pixelwise aligned) genuine documents, an unsupervised learning algorithm is employed to compute the document signature. The major feature used is the difference in alignment between genuine and fake documents. When an unknown document is presented, an attempt is made to match its signature with the genuine signature; the document is deemed to be fake if the signature does not match. The main assumption of this approach is that all genuine documents must be of the same type (e.g., like a particular bill or receipt) and a sufficient number of genuine samples must be available. Also, the approach assumes that a forged document is created using a scanner and printer, which introduce distortions.

2.2 Linking Documents to Source Devices

After a document is deemed to have been forged, it is often necessary to conduct a forensic investigation to discover its source. This is a hard problem.

One approach is to use a watermarking scheme in which a predetermined watermark is embedded in the document. The watermark is usually invisible to the naked eye but can be seen under a microscope. Depending on the level of sophistication, the watermark can help identify the make, model and even the specific printing device. However, a major shortcoming of this approach is that it relies on security through obscurity – the watermarking scheme must be kept secret to prevent unauthorized parties from creating the watermarks themselves.

Another approach is to characterize the defects that are unique to a particular scanner or printer. In this case, the amount of quantization done by a scanner, which differs from scanner to scanner, is used to link a fake document to a specific scanner [1, 4]. The unique color count is often used to link a forged document to a printer. Morphological features are also used to identify a printer [5]. The main advantages of this approach are that it does not rely on embedded watermarks and can be applied to any document.

Most approaches focus on detecting forgeries created by scanners and printers. However, high quality counterfeit currency is rarely, if ever, produced by scanners and printers. Indeed, high quality counterfeits are created using the same raw materials and printing processes used

Table 1. Mapping of inset letters to printing presses.

Inset Letter	Printing Press
Nothing, A, B, C, D	Press 1 in City 1
E, F, G, H, K	Press 2 in City 2
L, M, N, P, Q	Press 3 in City 3
R, S, T, U, V	Press 4 in City 4

to produce the originals. Therefore, it is extremely important to link counterfeit currency notes to their source printing presses. To the best of our knowledge, no research has specifically focused on this problem.

3. Currency Note Database

This section describes the database used for Indian currency note analysis. First, an overview is provided of Indian currency notes. Next, the genuine and counterfeit currency samples used in this work are described. Finally, the approach used to create digitized samples is detailed.

3.1 Indian Currency Notes

Indian currency notes are printed by the Reserve Bank of India (RBI) at four authorized currency presses. The presses are located in four different cities in India; the names of the cities are not publicized for security reasons [2].

An inset letter – a capital letter found on the number panel on the top right or bottom left of a currency note – is used to identify the printing press. Each of the four presses is allocated a set of inset letters for identification purposes. According to Gupta, *et al.* [2], 20 letters are currently used as inset letters. Table 1 presents the mapping of inset letters to printing presses. This information was inferred by Gupta, *et al.* [2] based on the name of the printer that appears on reams of printed banknotes.

This information is used as the ground truth in our experiments. Our goal is to attempt to construct a classifier for genuine Indian Rs. 500 notes. The classifier should partition input notes according to their source printing press.

3.2 Currency Samples

No publicly available database of genuine and counterfeit Indian currency notes currently exists. Furthermore, according to RBI regulations, no high resolution images of Indian currency notes may be publicly dis-

Table 2. Genuine currency samples.

Year	Inset Letter	RBI Governor	Series
2011	Nothing, E, L, R	Subbarao	Yellow
2010	Nothing, E, L, R	Subbarao	Yellow
2009	Nothing, E, L, R	Subbarao	Yellow
2008	Nothing, E, L, R	Reddy	Yellow
2007	E, L, R	Reddy	Yellow
2006	Nothing, E, L, R	Reddy	Yellow
2005	R	Reddy	Yellow
No Year	B	Reddy	Yellow
No Year	A	Jalan	Yellow
No Year	A	Jalan	Blue

seminated. Therefore, our only option was to create our own database of images.

We collected several Rs. 500 currency notes. In the case of genuine notes, we attempted to collect as many samples as possible of each type (i.e., year, inset letter and RBI Governor). The sample size was limited because some older series of notes were not readily available. Another limiting factor was cost. Our genuine currency sample set comprised three notes of each type listed in Table 2.

We were able to collect only ten counterfeit currency notes. Four of these notes were in very bad condition. Thus, the counterfeit currency sample set included just six notes.

3.3 Image Creation

High resolution images of the samples were created under different parts of the light spectrum. A visual spectral comparator (VSC 6000) with facilities for examining and photographing documents in varying lighting conditions was used for this purpose.

A total of 23 images were taken for each sample currency note. The images covered various parts of the notes under different lighting conditions and magnifications (Table 3). The imaging decisions were made based on a preliminary examination of the notes using the visual spectral comparator. The 23 images showed the most perceptible differences for the different currency notes examined. As such, they were assumed to be the most promising features for detecting counterfeits and identifying the source presses.

The digitized database thus consisted of 33 currency notes, 27 of them genuine and six counterfeit. Since 23 images were taken for each note, the database contained a total of 759 images.

Table 3. Features collected for each currency note.

Feature ID	Area of Note	VSC Setting	Mag.
IMG_01	Front, Entire note	Longpass = VIS	3.04
IMG_02	Front, Entire note	Longpass = RG925	3.04
IMG_03	Front, Entire note	IR, Longpass = RG630	3.04
IMG_04	Front, Entire note	IR, Longpass = OG530	3.04
IMG_05	Front, Entire note	UV 365nm	3.04
IMG_06	Front, Entire note	UV 312nm	3.04
IMG_07	Front, Entire note	UV 254nm	3.04
IMG_08	Front, Entire note	UV 365nm passthrough	3.04
IMG_09	Front, Entire note	Dim overhead light	3.04
IMG_10	Front center, Denomination in OVI	Longpass = VIS	16
IMG_11	Front center, Denomination in OVI	Longpass = VIS, Pseudocolor	16
IMG_12	Front center, Denomination in OVI	Longpass = RG925, Pseudocolor	16
IMG_13	Front, Hindi text RBI Governor	Longpass = VIS	61
IMG_14	Front, Braille identifier	Longpass = VIS	61
IMG_15	Front, Braille identifier	Longpass = VIS, Sidelight = Right	61
IMG_16	Front, Gandhi face	Longpass = VIS	25
IMG_17	Front, Gandhi ear, Micro-lettering	Longpass = VIS	30
IMG_18	Front, Inset letter	Longpass = VIS	50
IMG_19	Back, Entire note	Longpass = VIS	3.04
IMG_20	Back, Entire note	UV 365nm	3.04
IMG_21	Back, Entire note	UV 312nm	3.04
IMG_22	Back, Entire note	UV 254nm	3.04
IMG_23	Back, Entire note	UV 365nm passthrough	3.04

The database size was limited by the time constraints imposed by the digitizing process. It took an average of ten minutes to apply the required settings, focus, capture and save each image. Thus, it took about 230 minutes to fully digitize each sample. Due to the time factor, we decided to digitize one sample from each class (combination of year, inset letter and RBI Governor) of genuine notes along with the six counterfeit note samples. Thus, the database contained one representative sample from each series and type.

4. Detecting Counterfeit Currency Notes

This section describes the technique used to detect counterfeit currency notes, and the results that were obtained.

4.1 Preliminary Experiments

A preliminary experiment was conducted to determine which of the 23 features collected for each note would be useful in detecting counterfeit currency notes. Histograms of each feature were generated for all the currency notes and the histogram correlations of the corresponding features of the currency notes were examined. Also, the histograms were manually examined to discern differences that could assist in classifying notes as genuine or counterfeit. Since previous approaches successfully used the unique color count to detect fraudulent documents, the number of unique colors in each image was recorded.

The preliminary analysis also involved observations of the currency notes under a microscope. The Veho VMS-004D 400X USB microscope used for this purpose had fixed optical zooms of 20X and 400X. Various portions of the notes were examined for features that could be used to discriminate between genuine and fake notes.

The preliminary analysis revealed that features related to particular areas of a currency note were more discriminating that those related to the entire note. The most promising features observed were the Rs. 500 denomination lettering in optically variable ink, the area of Gandhi's face and the inset letter. The red, blue and green histograms of these features for genuine and counterfeit notes were compared, but no significant correlations were discerned. However, it appeared that there were clear differences between the unique color counts of features of genuine and fake currency notes, most likely due to differences in the printing process and ink.

4.2 Feature Selection

Three features were selected: (i) IMG_10 (Figure 1); (ii) IMG_11 (Figure 2); and (iii) IMG_12 (Figure 3). Based on the results of the preliminary experiments, the unique color count was used as a measure for quantifying the features. For each currency note, the corresponding images were selected and the total number of unique colors in the image was calculated. A C# program was written to perform this task and generate the corresponding CSV file of extracted features. It was not necessary to pre-process the features because they were already focused and adjusted at the time of sampling.

4.3 Classifier Design

A C4.5 decision tree was used to classify currency notes as genuine or counterfeit. The implementation of C4.5 in WEKA [6] was employed for

(a) Genuine note. (b) Counterfeit note.

Figure 1. Image IMG_10.

(a) Genuine note. (b) Counterfeit note.

Figure 2. Image IMG_11.

(a) Genuine note. (b) Counterfeit note.

Figure 3. Image IMG_12.

training and testing. A relatively simple classifier was preferred because the feature set comprised only three features.

We experimented with other classifiers, including neural networks, radial basis function networks and C4.5 using grafting. However, all four classifiers had comparable accuracy, so the classifier with the least complexity was selected. A simpler classifier requires less time for training and testing compared with a more complex classifier. In a real-world application involving counterfeit currency detection, it is necessary to provide a quick answer because the owner of a currency note would be unwilling to wait for a long period of time.

4.4 Results and Evaluation

A C4.5 decision tree was trained and tested using the database of 33 images (27 genuine notes and six counterfeit notes). Ten-fold stratified cross validation was employed; 90% of the set was used for training and the remaining 10% for testing. The number of correctly classified instances was 31 while the number of incorrectly classified instances was two, yielding an average accuracy of 93.94%. Of the two incorrectly classified instances, one was a false positive (genuine detected as fake) and the other was a false negative (fake detected as genuine).

Upon closer examination of the two incorrectly classified instances, we discovered that one was a genuine note that belonged to the old "Blue Series." This older series of Rs. 500 notes does not have as many security features as the newer notes, which may have led to it being erroneously classified as fake. Manual examination of the fake note that was classified as genuine revealed that the note had extensive markings (zigzag lines) in blue ink (possibly made with a ball point pen).

The two currency note samples were then removed and the classifier was tested once again. The results were 100% accurate with all 31 instances classified correctly.

5. Identifying the Source Printing Press

This section describes the technique used to link a currency note to its source printing press, and the results that were obtained.

5.1 Preliminary Experiments

Preliminary experiments were also conducted to identify the features that would help link a currency note to its source printing press. Only genuine notes were used in this experiment because the ground truth was known only for genuine notes. As mentioned earlier, notes with the same inset letter come from the same press. Consequently, the goal was

to successfully classify notes with the same inset letter and, thus, the same origin.

As in the previous experiment, histograms of the corresponding features of the notes were constructed. However, since there were more classes, it was difficult to identify one feature that could be used to differentiate between all the classes. Therefore, we attempted to discern some peculiar properties of each class of notes that would support the identification. We observed that, while no single feature was able to discriminate between all the classes, certain features were able to differentiate between one particular class and the others. Hence, we framed the task as a multilevel classification problem and used a cascade of classifiers. The cascade incorporates multiple levels of individual classifiers and passes the output of one classifier to another.

5.2 Pre-Processing

Pre-processing was required for some of the images before extracting the features. Two techniques were used for pre-processing:

- **Percentage Histogram Bins:** Histograms (bins 0–255) of the blue, green and red channels were constructed. The values of each bin were divided by the total pixels in the image and multiplied by 100 to expresses it as a percentage of pixels in the image with a particular intensity. This was necessary because the images had different sizes, which precluded the use of regular histograms for comparison (i.e., using only the numbers of pixels).

- **Threshold UV Images:** Threshold images were created from images obtained under ultraviolet (UV) light (Figure 4). Each pixel was given a value of 255 if its intensity in the green channel was greater than 100 and its intensity in the red channel was greater than 100; otherwise, it was given a value of zero. Thus, binary images were obtained with pixels of intensity zero or 255 (Figure 5).

5.3 Feature Selection

Based on the experimental results, we selected seven features to be used by the cascade of classifiers. Note that pre-processing was required for feature extraction and quantification.

Figure 4. UV image.

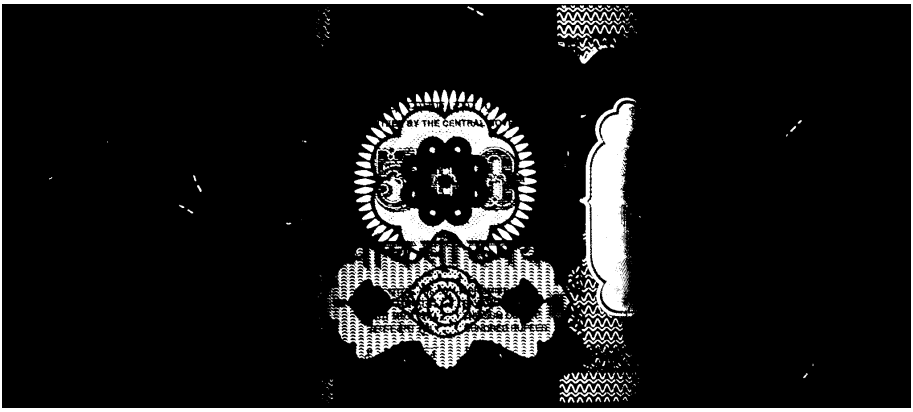


Figure 5. Threshold UV image.

- **Feat_01_Link:** The percentage histogram bins were calculated for IMG_18 (Figure 6). The quantification was performed using the sum of the histogram bins of the blue channel from 0 to 210.

- **Feat_02_Link:** The percentage histogram bins were calculated for IMG_18. The quantification was performed using the sum of the histogram bins of the blue channel from 0 to 100.

- **Feat_03_Link:** The percentage histogram bins were calculated for IMG_18. The quantification was performed using the sum of the histogram bins of the blue channel from 0 to 40.

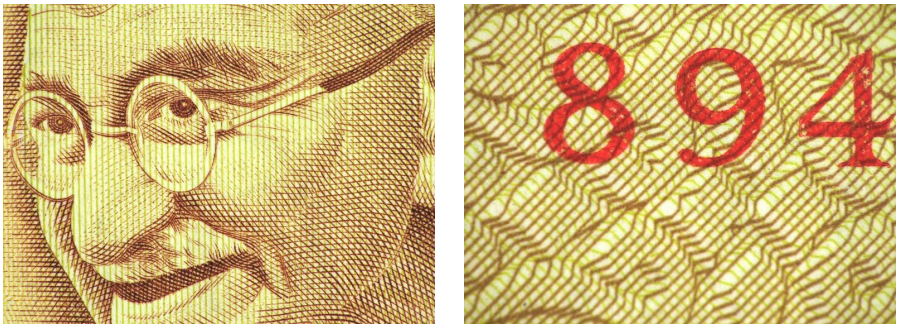


Figure 6. Images IMG_16 (left) and IMG_18 (right).

Figure 7. Image IMG_19.

- **Feat_04_Link:** The percentage histogram bins were calculated for IMG_16. The quantification was performed using the sum of the histogram bins of the green channel from 0 to 40.

- **Feat_05_Link:** The number of white pixels in each half of the threshold UV image of IMG_07 was computed. The quantification was performed using the ratio of the white pixels in the right half to those in the left half.

- **Feat_06_Link:** Image IMG_10 was used. The quantification was performed by counting the total number of unique colors.

- **Feat_07_Link:** Image IMG_19 was used (Figure 7). The quantification was performed by counting the total number of unique colors.

5.4 Classifier Design

As described above, a cascade of classifiers uses multiple levels of individual classifiers in which the output of one classifier is input to another. In our design, an individual classifier was used to filter currency notes belonging to one particular class. Thus, each succeeding classifier dealt with one less class than the preceding classifier in the cascade. Each level used a C4.5 decision tree as the classifier. A total of five classifiers were used.

- **Level 1:** This classifier was designed to filter the old Blue Series notes (i.e., Rs. 500 notes printed prior to or during Bimal Jalan's term as RBI Governor). These notes differ significantly from the newer Yellow Series notes. The old Blue Series notes have inset letters A, C or blank, and are from Press 1 in City 1. The features used in Level 1 were Feat_01_Link and Feat_02_Link. All the genuine notes were passed to the classifier, which classified them either as "Old Blue Series with inset letters A, C or nothing" or "New Yellow Series with any inset letter." The currency notes classified as "New Yellow Series" were passed to the Level 2 classifier.

- **Level 2:** This classifier was designed to filter notes with no inset letters. These notes are also from Press 1 in City 1. The feature used was Feat_03_Link. The output of the Level 1 classifier ("New Yellow Series") was passed as input to the Level 2 classifier, which classified the currency notes as "No inset letter" and "Inset letter A, B, E, L or R." The notes with inset letters A, B, E, L or R were passed to the Level 3 classifier.

- **Level 3:** This classifier was designed to filter notes with the inset letter L. These notes are from Press 3 in City 3. The feature used was Feat_04_Link. The output of the Level 2 classifier ("Inset letter "A, B, E, L or R") was passed as input to the Level 3 classifier, which classified the currency notes as "Inset letter L" and "Inset letter A, B, E or R." The notes with inset letters A, B, E or R were passed to the Level 4 classifier.

- **Level 4:** This classifier was designed to filter notes with the inset letter E. These notes are from Press 2 in City 2. The features used were Feat_05_Link and Feat_06_Link. Feat_05_Link was specifically used because our preliminary experiments revealed that the right portion of currency notes with the inset letter E had a larger area that glowed under UV light. The output of the Level 3 classifier ("Inset letter "A, B, E or R") was passed as input to the Level

Table 4. Results.

Level	Correctly Classified Instances	Incorrectly Classified Instances	Correctly Classified Percentage	Incorrectly Classified Percentage
Level 1	27	0	100%	0%
Level 2	26	0	100%	0%
Level 3	20	1	95.24%	4.76%
Level 4	14	1	93.33%	6.67%
Level 5	9	0	100%	0%

4 classifier, which classified the currency notes as "Inset letter E" and "Inset letter A, B or R." The notes with inset letters A, B or R were passed to the Level 5 classifier.

- **Level 5:** This classifier was designed to filter notes with the inset letter R. These notes are from Press 4 in City 4. Any note that was classified at this level was assumed to have inset letter A or B (from Press 1 in City 1). Feature Feat_07_Link was used because none of the other features could discriminate between notes with inset letters A, B and R. The output of the Level 4 classifier ("Inset letter "A, B or R") was passed as input to the Level 5 classifier, which classified the currency notes as "Inset letter R" and "Inset letter A or B."

5.5 Experimental Results

This section describes the results obtained for the individual classifiers and the cascade classifier.

- **Individual Classification:** In passing input to a classifier at a given level, we assumed that all the classifiers at the previous levels gave the correct results. Thus, the classifier input only contained instances that would be passed to it from the previous classifier. Ten-fold stratified cross validation was used for each classifier, except for the last (Level 5) classifier, which used eight-fold stratified cross validation.

  Table 4 shows the results that were obtained. Note that all five classifiers have high degrees of accuracy.

- **Cascaded Classification:** In this case, we evaluated the system of five cascaded classifiers as a whole. Each classifier was individually trained and then combined to create the cascade. We used

the entire set of currency notes to test the cascaded classifier – this was done to use all the available samples for testing and to see if the cascaded classifier failed on any sample.

A total of 27 samples were provided as input to the cascade classifier. Of these, 25 were classified correctly based on their inset letter and two were classified incorrectly. This corresponds to an overall accuracy of 92.59%.

5.6 Evaluation

While the experimental results indicate that the individual classifiers and the cascaded classifier have high degrees of accuracy, some limitations do in fact exist. First, the classifiers are dependent on the fact that the input images are correct and well focused. For example, the Level 2 classifier is sensitive to changes in focus. To verify this fact, we deliberately blurred a sharp image and provided it to the classifier, which gave an incorrect result.

The second limitation is that the currency notes are assumed to be of good quality. The presence of oil, cellotape, pen marks or dirt on the surface of a note can render it difficult to classify correctly. For example, one of the fake samples had blue pen marks over it, which caused it to be classified as genuine. Also, the presence of cellotape on a currency note produces an abnormal glow when viewed under UV light. The handling of such cases is important because many Indian currency notes are worn or soiled.

A third limitation is that, because the classifiers were trained with Rs. 500 notes, they cannot be applied to other denomination notes. Also, different denomination notes have different security features, and these differences have to be taken into account when training the classifiers.

6. Integrated Tool

The classifiers were implemented in an integrated tool, which was written in C#. The EmguCV library (a C# wrapper for OpenCV) was used to perform image processing operations. The tool was run on a Compaq Presario laptop with a 2 GHz Intel Core 2 Duo Processor T5800 and 2 GB RAM. The tool took as input the folder containing all 23 image features of the sample currency notes and classified each note as genuine or counterfeit. In the case of a genuine note, the tool also attempted to identify its source printing press. The evaluation of each note was completed within five seconds.

Most individuals do not have the expertise to manually examine a currency note and determine if it is counterfeit. The tool, especially one with an enhanced GUI, would be very useful to individuals who do not have much technical and forensic knowledge. Furthermore, the tool could be integrated with a USB microscope and scanner, which would greatly reduce the possibility of commercial establishments accepting counterfeit currency as payment for goods and services.

7. Conclusions

The single classifier approach described in this paper is well suited to detecting counterfeit currency notes. The cascaded classifier approach for linking genuine currency notes to their source printing presses is also fast and accurate. The prototype tool, which integrates the two classification approaches, functions as a standalone system for counterfeit currency detection and source press identification.

Our future research will expand the image database and test the classification approaches on samples of different denominations. Additionally, we will extend the identification approach to link counterfeit currency notes to their source printing presses in hostile states. Our ultimate goal is to develop a versatile and inexpensive tool that would enable individuals without much technical and forensic knowledge to quickly detect counterfeit currency notes and identify their source presses.

References

[1] C. Chang, T. Yu and H. Yen, Paper currency verification with support vector machines, *Proceedings of the Third IEEE International Conference on Signal-Image Technologies and Internet-Based Systems*, pp. 860–865, 2007.

[2] S. Gupta, D. Handa, R. Singh and K. Kumar, Forensic identification of Rs. 1,000 – Awareness for genuineness, *CBI Bulletin*, pp. 41–45, July-September 2011.

[3] G. Gupta, C. Mazumdar, M. Rao and R. Bhosale, Paradigm shift in document related frauds: Characteristics identification for development of a non-destructive automated system for printed documents, *Digital Investigation*, vol. 3(1), pp. 43–55, 2006.

[4] G. Gupta, S. Saha, S. Chakraborty and C. Mazumdar, Document frauds: Identification and linking fake documents to scanners and printers, *Proceedings of the International Conference on Computing Theory and Applications*, pp. 497–501, 2007.

[5] G. Gupta, R. Sultania, S. Mondal, S. Saha and B. Chanda, A structured approach to detect a scanner-printer used in generating fake documents, *Proceedings of the Third International Conference on Information Systems Security*, pp. 250–253, 2007.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, The WEKA data mining software: An update, *ACM SIGKDD Explorations Newsletter*, vol. 11(1), pp. 10–18, 2009.

[7] M. Ionescu and A. Ralescu, Fuzzy Hamming distance based banknote validator, *Proceedings of the Fourteenth IEEE International Conference on Fuzzy Systems*, pp. 300–305, 2005.

[8] C. Liu, S. Ruan, G. Huang, Y. Jian and L. Zhang, Research on identification of counterfeits by recognizing infrared images, *Proceedings of the International Conference on Microwave and Millimeter Wave Technology*, vol. 4, pp. 2081–2084, 2008.

[9] S. Ryu, H. Lee, I. Cho and H. Lee, Document forgery detection with SVM classifier and image quality measures, *Proceedings of the Ninth Pacific Rim Conference on Multimedia*, pp. 486–495, 2008.

[10] J. van Beusekom, F. Shafait and T. Breuel, Document signature using intrinsic features for counterfeit detection, *Proceedings of the Second International Workshop on Computational Forensics*, pp. 47–57, 2008.

[11] J. Xie, C. Qin, T. Liu, Y. He and M. Xu, A new method to identify the authenticity of banknotes based on the texture roughness, *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pp. 1268–1271, 2009.

[12] C. Yeh, W. Su and S. Lee, Employing multiple-kernel support vector machines for counterfeit banknote recognition, *Applied Soft Computing*, vol. 11(1), pp. 1439–1447, 2011.

Chapter 25

TOWARDS ACTIVE LINGUISTIC AUTHENTICATION

Patrick Juola, John Noecker Jr., Ariel Stolerman, Michael Ryan, Patrick Brennan and Rachel Greenstadt

Abstract Biometric technologies offer a new and effective means for securing computers against unauthorized access. Linguistic technologies and, in particular, authorship attribution technologies can assist in this effort. This paper reports on the results of analyzing a novel corpus that was developed to test the possibility of active linguistic authentication. The study collected the one-week work product of nineteen temporary workers in a simulated office environment. The results demonstrate that techniques culled from the field of authorship attribution can identify workers with more than 90% accuracy.

Keywords: Stylometry, authentication, authorship attribution, profiling

1. Introduction

Standard password-based identification systems are known to have flaws. Passwords can be forgotten, written down, stolen, and guessed. If any of these events occur, an intruder has the keys to the kingdom. Biometric-based identification systems have been proposed to bypass or mitigate some of these problems – it is hard to forget your own thumbprint. But developing and testing these systems can be a challenge precisely because of the need to handle a wide variety of humans, especially when the biometric task is challenging and time-consuming.

One possibility for biometric validation is the individual use of language. Prior work has shown that the authorship of documents containing as little as a few hundred words can be correctly identified. In a typical office environment, a worker types many more words each day and, thus, could continually identify himself or herself to an appropriate security screening program. This paper reports on the development of

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics IX, IFIP AICT 410, pp. 385–398, 2013.

a novel corpus to enable this type of analysis and provides the results of a proof-of-concept analysis.

2. Background

This section discusses authentication, stylometry and authorship attribution, and the JGAAP and JStylo systems.

2.1 Authentication

Passwords are commonly used to secure computer systems. The flaws associated with passwords are well known. Passwords can be forgotten, which prevents access. Because passwords are difficult to remember, they are often written down, but this means that they can be stolen and used to enable unauthorized access. Also, passwords can be guessed or recovered using cryptanalytic techniques.

More subtly, traditional passwords are limited in the type of protection they provide. Once a person presents his password, he is authenticated and may use his computer. If this person steps away from his desk for a moment, anyone could step up to the keyboard and use the computer. Chaski [5] defined this as the "who's at the keyboard" dilemma and cites several examples, including the case of a dead body found in a room with a disputed suicide note typed on a computer, and the case of an email sent while an employee was away from her desk at lunch. Coulthard [6] describes a similar case involving a disputed email. In other words, traditional password protection provides only passive "perimeter" security that does not prevent "insiders" from abusing their status.

An improved security model would involve continuous, active authentication where the behavior of the person at the keyboard is monitored. Security measures can be triggered immediately when the behavior of the person changes. The security measures can be as simple as reauthentication or as complicated as triggering a call to security personnel and locking the computer down.

2.2 Stylometry and Authorship Attribution

Authorship attribution, also called stylometry or stylistics, is a well-established field of study [10, 12, 18, 24], although it has typically not been used for authentication. The theory behind authorship attribution is that each person has his own unique "stylome" [28], a unique set of idiolectal choices that describe his speaking and writing style. At a group level, this is the kind of choice that causes the British to walk on "pavements" instead of "sidewalks." At an individual level, it is

the kind of choice that causes a person to place a fork "to" the left of the plate instead of "on" the left or "at" the left. Quantifying these choices, for example, by making a histogram of function words [3, 21] or of character n-grams [25] can enable investigators to develop a computationally tractable summary of stylistic choices and form judgments based on these summaries.

Standard practice in stylometric investigations involves a detailed comparison of stylistic features culled from a training set of documents. The questioned document is then compared against the training set, typically using a classification or machine learning algorithm, and an appropriate decision is taken. A related problem is authorship verification, where a novel document is compared with a summary by a single author. If the novel document is close (stylistically) to the summary, it is inferred to been written by the author; and if it is distant, then not.

A third application of stylometric technology is in stylistic profiling [2, 8, 16, 19, 27], where the objective is not to identify a specific person, but to identify characteristics of the writer such as age, gender, social class and native language. Again, a typical study involves collecting samples of (for example) male writing and female writing, and then comparing a questioned document against the stylistic summaries to classify the document as being written by a male or female.

The application of this technology to authentication is straightforward. Instead of using a training set of documents, a pseudo-document containing the user's long-term behavior is used, and the recent behavior at the keyboard is verified as being consistent with the long-term behavior. A significant inconsistency would trigger a security response. We are, therefore, proposing the use of linguistic stylistics as a biometric, similar to the use of typing speed or mouse movements [30].

As a purely text classification technology (i.e., not real time and not using keystrokes, but instead using finished documents and often involving forced-choice comparisons between a fixed group of authors), authorship attribution is in some regards a mature technology. For example, at the PAN-2012 Conference (see `pan.webis.de`), the top three methods all classified more than 80% of 241 documents correctly with (in some cases) more than a dozen distractor authors. We have reason to believe that authorship authentication may be even more accurate because issues such as automatic spelling correction [6] will not normalize individual writing patterns – someone who is a poor typist or who continually misspells "*toutch" [29] will not have this idiosyncratic quirk airbrushed away. We expect that an appropriately chosen analytic method will eventually achieve similar or better results in this novel context.

2.3 JGAAP

In light of the differences among possible analyses, an obvious question is: Which method works best? In order to address this question, the Evaluating Variations in Language Laboratory at Duquesne University has developed a modular system for the development and comparative testing of authorship attribution methods [12, 15]. This system, called JGAAP (Java Graphical Authorship Attribution Program) provides a large number of interchangeable analysis modules to handle different aspects of the analysis pipeline such as document preprocessing, feature selection and analysis/visualization.

The JGAAP project has been very successful, creating one of the most widely-used systems for authorship analysis, leading the way in the search for best practices, and developing a group of protocols accurate enough to have been used in court [14]. Most importantly, it has created a standard, tested set of operational primitives (such as approximately two dozen ways to assess linguistic differences) [26] based on various underlying cognitive models and computational approaches [11]. This toolset will be leveraged in a wide-ranging and systematic exploration of several different types of analysis and relationships. Taking combinatorics into account, the number of different ways to analyze a set of documents numbers in the millions and this can be expanded by an inventive user. JGAAP is freely available (from `www.jgaap.com`), making it a useful testbed for other researchers.

For example, Grant [7] describes a criminal case involving vocabulary comparisons among text messages sent by a number of people. The technical question involved in this case hinged on the existence and number of specific words (or spelling variants such as "wen" for "when" or "4get" for "forget") that were used by the one person who was involved. This can be captured by measuring document similarity using the Jaccard or intersection distance, essentially a measure of vocabulary overlap without regard to specific frequencies. In contrast, the classic Mosteller-Wallace [21] study of historical documents examined frequency differences among common (and therefore shared) vocabulary. The important question was not whether or not people used words like "upon" (because we all do), but whether the disputed document used that word more like person A or person B. This type of analysis can be done by measuring document similarity using frequency measures such as normalized cosine (dot-product) distance [22] or Manhattan distance. JGAAP has been expanded to include all three of these measures as well as many others.

2.4 JStylo

JStylo (see `psal.cs.drexel.edu`) is an open-source authorship attribution platform developed at the Privacy, Security and Automation Laboratory at Drexel University on top of the JGAAP project. It was primarily created to allow cross-feature analysis, where multiple features can be extracted and included in one analysis, an option that was not available in JGAAP at the time. JStylo is complemented by a dual analysis tool, along with Anonymouth [20], a writing-style anonymization platform, whose underlying authorship attribution engine is JStylo.

In JStylo, every feature can be one of two types: a class of feature frequencies (e.g., features "a"..."z" for the "Letters" feature class), or a numeric evaluation of the input documents (e.g., Yule's characteristic K). An additional advantage of JStylo is its fine-resolution feature definition capabilities. Each feature is uniquely defined by a set of its own document-preprocessing tools, one unique feature extractor (core of the feature), feature-postprocessing tools and normalization/factoring options. All of JGAAP core features are available in JStylo, in addition to some newly developed features such as regular-expression-based extraction.

With regard to analysis capabilities, the main classification tools available in JStylo are drawn from Weka [9], the popular data mining and machine learning platform. These include classifiers commonly used for authorship attribution such as support vector machines, neural networks, naïve Bayes classifiers and decision trees. In addition, JStylo provides an implementation of the Writeprints authorship attribution technique [1], which is known for its high accuracy in scenarios with large numbers of authors.

Although JStylo lacks the maturity of JGAAP, it compensates with a vast range of features and, more importantly, the ability to combine them. This capability was leveraged to conduct the preliminary analysis described in this paper, where the extensive feature set used with the Writeprints method was applied to the collected data.

3. Work Product Corpus

We created a simulated work environment to generate a suitable corpus for validation. A rented space in downtown Pittsburgh was set up as an office staffed by temporary employees (subjects). The subjects were supervised by Juola & Associates staff and asked, over the course of a week, to research and write blog articles "related to Pittsburgh in some way." This provided them with an incentive to use standard computer tools such as browsers and search engines to conduct research and word

processors to do the actual writing. The task was expected to take approximately six hours per day, except for a shorter first day as described below. The subjects were provided with a reasonable degree of topical similarity, but they had enough freedom to be individually distinctive so that they could not be trivially distinguished on the basis of the type of task they were doing. The subjects were not restricted from accessing personal websites or playing standard games, and they were allowed to copy and paste material as long as the final articles were their own work. As expected, the most commonly-used applications by the subjects were Internet Explorer and Microsoft Word.

The subjects were also advised that their computers were equipped with tracking software, in particular, a macro recorder for measuring keyboard use, including individual keystrokes and dynamic information such as timing, length of keypress and overlap between keys. The macro recorder also measured mouse events such as clicks and movements. Keylogging software was used to record text as it was entered, including mapping text to specific applications, clipboard use and browsing history. The subjects were notified that, although good faith efforts would be made to scrub the data prior to analysis, all input would be captured (including personal information such as Facebook account names and passwords) and that it could not be guaranteed that all this information would be wiped after the experiments were completed. The subjects were given an opportunity to request that specific strings (e.g., user names and passwords) be automatically redacted, but it would have been easier for subjects just to change their Facebook passwords or not log into Facebook from work.

In addition to the main tasks, the subjects had two types of secondary tasks. The morning of the first day was spent in an orientation process that included the administration of a number of psychometric tests that measured traits such as personality, self-esteem and learning styles. (We do not report on this aspect of the study.) During the final two hours of the day, the subjects were asked to perform a set of small, explicitly-defined tasks (microtasks) such as describing a specific local landmark or event or summarizing an article. This provided a set of very detailed, task-specific data with much tighter control, possibly creating a different environment for task-focused inter-individual comparisons.

Data collection is ongoing. By the end of the project, we expect to have the work product of at least 80 subjects.

4. Analysis

Our analysis is based on data gathered over three weeks according to the protocol described in the previous section. The data set comprised five days of work for each of fourteen subjects (one subject failed to show up for work after being hired). One day of work for one participant was temporarily mislaid; this has since been addressed, but the work product was not analyzed. Consequently, we have a total of 69 days of work product. This work product comprised approximately 280 MB of data, including 17.5 million mouse and keyboard events and 23,000 website visits. Our analysis in this paper only focuses on the language used in the keystroke events.

4.1 Daily Data

In the first phase, we analyzed each day of work as a unit, using hold-one-out cross-validation (i.e., each document was analyzed individually against the other 68 documents) in a content-agnostic way using only character n-gram distribution frequencies. We recognize that requiring a full day of work prior to making security decisions is impractical, but this provides a baseline against which smaller samples can be measured.

Our analysis was performed using JGAAP with three canonicizers: (i) Unify Case, which neutralizes all case distinctions; (ii) Normalize Whitespace, which replaces tabs, newlines and multiple spaces with a single space character; and (iii) Keylogger, which cleans up the logs in several ways. The Keylogger canonicizer removed anything that did not represent a keystroke, including time/date stamps from the keylogger, information about the window from which keystrokes originated, and whitespace to make the logs readable. Note that this means that if a subject typed something in a browser and then switched to Word, there would be nothing left in the log to record this activity. Thus, we would have "google.com-ENTER-is a PittsburghHotels in Pittsburgh institution" (i.e., multiple window inputs mashed together).

Next, we converted special keys to single character representations. For example, -ENTER- was converted to a newline. Or, arrow key -UP- was was converted to a placeholder that was unlikely to appear in the actual input.

Finally, the analysis was performed using a simple nearest-neighbor classifier with the Manhattan distance (i.e., L_1 distance) or intersection distance (i.e., Jaccard distance) based on histograms of character n-grams of lengths ranging from one through five.

The results are shown in Table 1. All results are based on the number of definitive classifications, nominally 69 documents (denominator).

Table 1. Daily analysis classification results.

Analysis Method	Results
Manhattan 1-grams	37/69
Manhattan 2-grams	53/69
Manhattan 3-grams	62/69
Manhattan 4-grams	58/69
Manhattan 5-grams	50/69
Intersection 1-gram	6/21
Intersection 3-gram	23/68
Intersection 4-gram	22/69
Intersection 5-gram	22/69

However, all ties are reported as a single author, which yields a lower number in the denominator (e.g., 21 and 68).

Based on these results, it is clear that individual subjects can be distinguished with high accuracy. The result is 88.4% accurate, better, in fact, in purely nominal terms than the PAN-2012 Conference winner. It is also clear that, in this particular framework, the Manhattan distance is a more promising and accurate measure than the intersection distance, suggesting that it is more useful to measure frequency differences than mere presence/absence distinctions. Nevertheless, the fact that decisions were possible at all in more than 21 cases using individual characters and intersection distance hints at the power of using keyboard interactions in a forensic or security tool. In the 21 cases, there were certain keys that some individuals did not hit at all over the course of an entire day. Clearly, this is a much richer set of features and events than just alphanumeric characters and punctuation.

4.2 Fixed-Size Sliding Window

In the second phase of the analysis, we concatenated the keystroke data of all the users and re-divided the result into consecutive non-overlapping documents (windows) with predefined fixed sizes of 100, 500 and 1,000 words. This type of analysis is closer to the active authentication problem we aim to solve, because any real-time monitoring system would eventually be based on evaluating sliding windows of user input on-the-fly in an attempt to catch unauthorized users. One of the challenges is to decrease the window size as much as possible (leading to a quicker response time) while maintaining high accuracy and low false positives and false negatives (i.e., undetected unauthorized users and false alarms for authorized users, respectively).

Table 2. Writeprints-inspired feature set.

Group	**Features**
Lexical	Character count
	Average word length
	Letters
	50 most common letter bigrams
	50 most common letter trigrams
	Percentage of letters
	Percentage of uppercase letters
	Percentage of digits
	Digits
	2-digit numbers
	3-digit numbers
	Word length distribution
	Special characters
Syntactic	50 most common function words
	Punctuation
	Part-of-speech (POS) tags
	50 most common POS bigrams
	50 most common POS trigrams
Content	50 most common words in the corpus
	50 most common word bigrams in the corpus
	50 most common word trigrams in the corpus
Idiosyncrasies	Common misspellings

As in the first phase, the data was stripped of keylogger metadata, special keys were converted to unique single-character placeholders and whitespace was normalized. However, the raw data was not case unified and all special keys (e.g., -ENTER- and -TAB-) were replaced with placeholders (rather than being converted to newline and tab, respectively) in order to preserve user typing characteristics to the extent possible. Since the data includes special characters, it is more accurate to measure document length in terms of tokens than words (e.g., ch$\beta\beta$Cch$\beta\beta$hicago where β represents backspace).

The second phase used a different feature set from the first phase. Specifically, a close variation of the Writeprints [1] feature set was used; this set includes a vast range of linguistic features across different levels of text (summarized in Table 2). The rich linguistic feature set better captures user writing styles. With the help of the special-character placeholders, some features capture aspects of user style that are usually not found in standard authorship problem settings. For example,

Table 3. Sliding-window analysis classification results.

Window Size	Accuracy	Weighted Avg. FN	Weighted Avg. FP
SMO			
100	81.07%	18.93%	2.0%
500	93.06%	6.94%	0.9%
1,000	93.33%	6.77%	0.9%
KNN			
100	71.79%	28.21%	2.4%
500	83.04%	16.96%	1.5%
1,000	83.13%	16.87%	1.1%

frequencies of backspaces and deletes provide some evaluation of a user's typo rate or lack of decisiveness.

Our analysis was performed in JStylo using 10-fold cross-validation for evaluation. We used two Weka classifiers: KNN classifier with $K = 1$ and Manhattan distance (similar to the previous phase) and SMO SVM [23] with a soft margin constant $C = 1$ and polynomial kernel of degree one. SMO solves multi-class problems using pairwise binary classification. The features are normalized by default for both classifiers.

The results for the second phase are shown in Table 3. If it was clear from the first phase that the subjects can be distinguished with high accuracy based on one day of work product, the results in the second phase demonstrate that high distinguishability can be achieved by examining token sequences of up to 1,000 in length. Moreover, the statistically insignificant ($p < 0.01$) difference between the results for 500-token and 1,000-token windows implies that verification could be achieved after merely 500 tokens of user input. However, the statistically significant ($p < 0.01$) difference in accuracy when dropping down to 100-token windows suggests that there is a minimal threshold to consider for these settings. Finally, support vector machines, which are used extensively in authorship attribution due to their high performance and accuracy prevailed in these settings as well. In particular, the support vector machines outperformed KNN ($p < 0.01$), with the best results being 93.33% accuracy for 1,000-token windows and 93.06% accuracy for 500-token windows.

5. Conclusions

From a practical standpoint, waiting for almost an entire day to determine whether or not an unauthorized individual is using a work com-

puter leaves much to be desired. Nevertheless, the results demonstrate the success of the proof-of-concept system and that it is easy enough to vary the window size and other parameters to obtain good results. For example, it could be useful to compare and analyze the accuracy achieved with temporal windows that consider hour-long, five-minute long or minute-long slices of work instead of length-based windows.

Similarly, we have chosen only a few types of features/events to analyze out of the dozens provided by JGAAP and JStylo, just a few types of classifiers and ignored the possibility of using other classification techniques such as binary-class support vector machines, neural networks, linear discriminant analysis and latent Dirichelet allocation. In the longer run, it has been shown that ensemble methods like mixture-of-experts [13] tend to outperform individual analyses, and it is necessary to investigate that, even if small/short samples do not work well under single analysis, it is possible to achieve good authentication with multiple independent analyses. Also, it may be promising to combine linguistic biometrics with other data sources (e.g., mouse movements that have yielded good results [30]) as a base for authentication.

From a security standpoint, a key question is accuracy in the face of active deception that fools the monitor. While our experiments do not consider this issue, recent research in stylistic deception [4, 17, 20] provides avenues for future work in this area.

The U.S. Department of Defense has suggested (in DARPA BAA 12-06) that computer-captured biometrics can be "used to uniquely recognize humans" with high accuracy and minimal intrusiveness. We believe that our work confirms this suggestion, providing approximately 90% accuracy across nearly two dozen subjects. While further work is required to improve accuracy and to address verification in unknown settings – and possibly to integrate with other sources of biometric information and to develop a commercial-scale security system – the results presented in this paper strongly confirm the promise of our approach.

Acknowledgement

This research was supported by the National Science Foundation under Grant No. OCI-1032683 and by DARPA under BAA-12-06.

References

[1] A. Abbasi and H. Chen, Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace, *ACM Transactions on Information Systems*, vol. 26(2), pp. 7:1–7:29, 2008.

[2] S. Argamon, M. Koppel, J. Pennebaker and J. Schler, Automatically profiling the author of an anonymous text, *Communications of the ACM*, vol. 52(2), pp. 119–123, 2009.

[3] J. Binongo, Who wrote the 15th Book of Oz? An application of multivariate analysis of authorship attribution, *Chance*, vol. 16(2), pp. 9–17, 2003.

[4] M. Brennan and R. Greenstadt, Practical attacks against authorship recognition techniques, *Proceedings of the Twenty-First Conference on Innovative Applications of Artificial Intelligence*, pp. 60–65, 2009.

[5] C. Chaski, Who's at the keyboard: Authorship attribution in digital evidence investigations, *International Journal of Digital Evidence*, vol. 4(1), 2005.

[6] M. Coulthard, On the admissibility of linguistic evidence, *Brooklyn Law School Journal of Law and Policy*, vol. 21(2), pp. 441–466, 2013.

[7] T. Grant, TXT 4N6: Method, consistency and distinctiveness in the forensic authorship analysis of SMS text messaging, *Brooklyn Law School Journal of Law and Policy*, vol. 21(2), pp. 467–494, 2013.

[8] C. Gray and P. Juola, Personality identification through on-line text analysis, presented at the *Chicago Colloquium on Digital Humanities and Computer Science*, 2012.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, The Weka Data Mining Software: An update, *SIGKDD Explorations Newsletter*, vol. 11(1), pp. 10–18, 2009.

[10] M. Jockers and D. Witten, A comparative study of machine learning methods for authorship attribution, *Literary and Linguistic Computing*, vol. 25(2), pp. 215–223, 2010.

[11] P. Juola, Operationalizing lexical choice in language change, presented at the *First Conference on Quantitative Investigation in Theoretical Linguistics*, 2002.

[12] P. Juola, Authorship attribution, *Foundations and Trends in Information Retrieval*, vol. 1(3), pp. 233–334, 2008.

[13] P. Juola, Authorship attribution: What mixture-of-experts says we don't yet know, presented at the *American Association for Corpus Linguistics Conference*, 2008.

[14] P. Juola, Authorship and immigration: A case study, *Brooklyn Law School Journal of Law and Policy*, vol. 21(2), pp. 287–298, 2013.

[15] P. Juola, J. Noecker, M. Ryan and S. Speer, JGAAP 4.0 – A revised authorship attribution tool, presented at the *Digital Humanities Conference*, 2009.

[16] P. Juola, M. Ryan and M. Mehok, Geographically localizing tweets using stylometric analysis, presented at the *American Association for Corpus Linguistics Conference*, 2011.

[17] P. Juola and D. Vescovi, Empirical evaluation of authorship obfuscation using JGAAP, *Proceedings of the Third ACM Workshop on Artificial Intelligence and Security*, pp. 14–18, 2010.

[18] M. Koppel, J. Schler and S. Argamon, Computational methods in authorship attribution, *Journal of the American Society for Information Science and Technology*, vol. 60(1), pp. 9–26, 2009.

[19] M. Koppel, J. Schler and K. Zigdon, Determining an author's native language by mining a text for errors, *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 624–628, 2005.

[20] A. McDonald, S. Afroz, A. Caliskan, A. Stolerman and R. Greenstadt, Use fewer instances of the letter "i": Toward writing style anonymization, in *Privacy Enhancing Technologies*, S. Fischer-Hubner and M. Wright (Eds.), Springer-Verlag, Berlin, Germany, pp. 299–318, 2012.

[21] F. Mosteller and D. Wallace, *Inference and Disputed Authorship: The Federalist*, Addison-Wesley, Reading, Massachusetts, 1964.

[22] J. Noecker and P. Juola, Cosine distance nearest-neighbor classification for authorship attribution, presented at the *Digital Humanities Conference*, 2009.

[23] J. Platt, Fast training of support vector machines using sequential minimal optimization, in *Advances in Kernel Methods: Support Vector Learning*, B. Scholkopf, C. Burges and A. Smola (Eds.), MIT Press, Cambridge, Massachusetts, pp. 185–208, 1999.

[24] E. Stamatatos, A survey of modern authorship attribution methods, *Journal of the American Society for Information Science and Technology*, vol. 60(3), pp. 538–556, 2009.

[25] E. Stamatatos, On the robustness of authorship attribution based on character n-gram features, *Brooklyn Law School Journal of Law and Policy*, vol. 21(2), pp. 421–439, 2013.

[26] S. Stein and S. Argamon, A mathematical explanation of Burrows's delta, presented at the *Digital Humanities Conference*, 2006.

[27] H. van Halteren, Author verification by linguistic profiling: An exploration of the parameter space, *ACM Transactions on Speech and Language Processing*, vol. 4(1), pp. 1:1–1:17, 2007.

[28] H. van Halteren, R. Baayen, F. Tweedie, M. Haverkort and A. Neijt, New machine learning methods demonstrate the existence of a human stylome, *Journal of Quantitative Linguistics*, vol. 12(1), pp. 65–77, 2005.

[29] F. Wellman, *The Art of Cross-Examination*, Macmillan, New York, 1936.

[30] N. Zheng, A. Paloski and H. Wang, An efficient user verification system via mouse movements, *Proceedings of the Eighteenth ACM Conference on Computer and Communications Security*, pp. 139–150, 2011.