



Advances in Digital Forensics II

Edited by
Martin Olivier
Sujeet Shenoi



Springer



ifip

ADVANCES IN DIGITAL FORENSICS II

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 less 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 in information 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.

ADVANCES IN DIGITAL FORENSICS II

*IFIP International Conference on Digital
Forensics, National Center for Forensic Science,
Orlando, Florida, January 29- February 1, 2006*

Edited by

Martin S. Olivier

University of Pretoria, Pretoria, South Africa

Sujeet Sheno

University of Tulsa, Tulsa, Oklahoma, USA



Springer

Library of Congress Control Number: 2006929911

Advances in Digital Forensics II

Edited by M. Olivier and S. Sheno

p. cm. (IFIP International Federation for Information Processing, a Springer Series in Computer Science)

ISSN: 1571-5736 / 1861-2288 (Internet)

ISBN: 10: 0-387-36890-6

ISBN: 13: 9780-387-36890-6

eISBN: 10: 0-387-36891-4

Printed on acid-free paper

Copyright © 2006 by International Federation for Information Processing.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

9 8 7 6 5 4 3 2 1

springer.com

Contents

Contributing Authors	ix
Preface	xvii
PART I THEMES AND ISSUES	
1	
Some Challenges in Digital Forensics	3
<i>Eugene Spafford</i>	
PART II EVIDENCE COLLECTION AND HANDLING	
2	
AFF: An Open Extensible Format for Disk Imaging	13
<i>S. Garfinkel, D. Malan, K. Dubec, C. Stevens and C. Pham</i>	
3	
File System Support for Digital Evidence Bags	29
<i>Golden Richard III and Vassil Roussev</i>	
4	
Remote Upload of Evidence Over Mobile Ad Hoc Networks	41
<i>Indrajit Ray</i>	
5	
Applying Machine Trust Models to Forensic Investigations	55
<i>M. Wojcik, H. Venter, J. Eloff and M. Olivier</i>	
6	
Exploring Big Haystacks: Data Mining and Knowledge Management	67
<i>Mark Pollitt and Anthony Whitledge</i>	

PART III FORENSIC TECHNIQUES

7		
Countering Hostile Forensic Techniques		79
<i>Scott Piper, Mark Davis and Sujeet Sheno</i>		
8		
Using PLSI-U to Detect Insider Threats From Email Traffic		91
<i>James Okolika, Gilbert Peterson and Robert Mills</i>		
9		
Collusion Detection Using Multimedia Fingerprints		105
<i>Anthony Persaud and Yong Guan</i>		
10		
Authorship Attribution for Electronic Documents		119
<i>Patrick Juola</i>		
11		
Linking Individuals to Digital Information		131
<i>Shelly Seier, David Greer and Gavin Manes</i>		
12		
Use-Misuse Case Driven Analysis of Positive Train Control		141
<i>Mark Hartong, Rajni Goel and Duminda Wijesekera</i>		

PART IV OPERATING SYSTEM AND FILE SYSTEM FORENSICS

13		
Mac OS X Forensics		159
<i>Philip Craiger and Paul Burke</i>		
14		
Detecting Data Concealment Programs Using File System Analysis		171
<i>M. Davis, R. Kennedy, K. Pyles, A. Strickler and S. Sheno</i>		
15		
Assessing Trace Evidence Left by Secure Deletion Programs		185
<i>Paul Burke and Philip Craiger</i>		

PART V NETWORK FORENSICS

16	
On the Reliability of Network Eavesdropping Tools	199
<i>Eric Cronin, Micah Sherr and Matthew Blaze</i>	
17	
Active Traffic Capture for Network Forensics	215
<i>Marco Slaviero, Anna Granova and Martin Olivier</i>	
18	
Logical Traffic Isolation Using Differentiated Services	229
<i>Tinus Strauss, Martin Olivier and Derrick Kourie</i>	
19	
Passive Detection of NAT Routers and Client Counting	239
<i>Kenneth Straka and Gavin Manes</i>	
20	
Analysis of Web Proxy Logs	247
<i>B. Fei, J. Eloff, M. Olivier and H. Venter</i>	
21	
GSM Cell Site Forensics	259
<i>Christopher Swenson, Tyler Moore and Sujeet Sheno</i>	
22	
An Architecture for SCADA Network Forensics	273
<i>T. Kilpatrick, J. Gonzalez, R. Chandia, M. Papa and S. Sheno</i>	
PART VI PORTABLE ELECTRONIC DEVICE FORENSICS	
23	
Identifying Digital Cameras Using CFA Interpolation	289
<i>Sevinc Bayram, Husrev Sencar and Nasir Memon</i>	
24	
Forensic Analysis of BIOS Chips	301
<i>Pavel Gershteyn, Mark Davis and Sujeet Sheno</i>	

PART VII TRAINING, GOVERNANCE AND LEGAL ISSUES

25

- A Training Tool for Internet Crimes Against Children Cases 317
S. Aggarwal, B. Breeden, P. Henry and J. Mulholland

26

- Process Flow Diagrams for Training and Operations 331
Jacobus Venter

27

- A Control Framework for Digital Forensics 343
S. von Solms, C. Louwrens, C. Reekie and T. Grobler

28

- Criminal Regulation of Anti-Forensic Tools in Japan 357
Tetsuya Ishii

Contributing Authors

Sudhir Aggarwal is a Professor of Computer Science at Florida State University, Tallahassee, Florida. His research interests include building software tools and systems in support of digital forensics and law enforcement, and computer network issues in network games.

Sevinc Bayram is a Research Assistant in the Department of Electrical and Electronics Engineering, Uludag University, Bursa, Turkey. Her research interests are in the area of digital image forensics.

Matthew Blaze is an Associate Professor of Computer and Information Science at the University of Pennsylvania, Philadelphia, Pennsylvania. His research interests include secure systems and cryptography, trust management, and technology and public policy.

Bob Breeden is a Special Agent Supervisor at the Florida Computer Crime Center, Florida Department of Law Enforcement, Tallahassee, Florida. His research interests include all aspects of computer-related crime pertaining to enforcement and prevention.

Paul Burke is a Senior Digital Evidence Research Assistant at the National Center for Forensic Science, University of Central Florida, Orlando, Florida. His research interests include network security, digital forensics and open source operating systems.

Rodrigo Chandia is a Ph.D. student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include SCADA security, computer security and open source software development methodologies.

Philip Craiger is the Assistant Director for Digital Evidence at the National Center for Forensic Science and an Assistant Professor of Engineering Technology at the University of Central Florida, Orlando, Florida. His research interests include digital forensics and information security.

Eric Cronin is a Ph.D. candidate in Computer and Information Science at the University of Pennsylvania, Philadelphia, Pennsylvania. His research interests include network security, privacy and distributed systems.

Mark Davis is a Ph.D. student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include digital forensics, security auditing and information assurance.

Karl-Alexander Dubec is an undergraduate student at Harvard University, Cambridge, Massachusetts.

Jan Eloff is a Professor of Computer Science and Chair of the School of Information Technology at the University of Pretoria, Pretoria, South Africa. His research interests include computer and information security.

Bennie Fei is an M.Sc. student in Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include information visualization for digital forensics.

Simson Garfinkel is a postdoctoral fellow at the Center for Research on Computation in Society at Harvard University and a consulting scientist at Basis Technology Corp., Cambridge, Massachusetts. His research interests include computer security and forensics.

Pavel Gershteyn is an undergraduate student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include digital forensics.

Rajni Goel is an Assistant Professor of Information Systems and Decision Sciences at Howard University, Washington, DC. Her research interests include information assurance and digital forensics.

Jesus Gonzalez is a Ph.D. student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include network and SCADA systems security, and intrusion detection.

Anna Granova is pursuing her LL.D. in Law at the University of Pretoria, Pretoria, South Africa. Her research interests include public international law and digital forensics.

David Greer is a Ph.D. student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include digital forensics and information assurance.

Talania Grobler is a Ph.D. student in Informatics Science at the University of Johannesburg, Johannesburg, South Africa. Her research focuses on the management of digital forensic investigations.

Yong Guan is an Assistant Professor of Electrical and Computer Engineering at Iowa State University, Ames, Iowa. His research interests include digital forensics, wireless and sensor network security, and privacy-enhancing technologies for the Internet.

Mark Hartong is a Senior Electronics Engineer with the Office of Safety, Federal Railroad Administration, U.S. Department of Transportation, Washington, DC. He is a Ph.D. student in Information Technology at George Mason University, Fairfax, Virginia. His research interests include software engineering, information security and forensics.

Peter Henry is a Ph.D. student in the College of Information at Florida State University, Tallahassee, Florida. His research interests include digital forensics and human-computer interface security.

Tetsuya Ishii is an Associate Professor in the Faculty of Law and Economics at Chiba University, Chiba, Japan. His research focuses on the legal aspects of information security and digital forensics.

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

Richard Kennedy is an undergraduate student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include computer networks, digital forensics and electronic commerce security.

Tim Kilpatrick is an M.S. student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include SCADA systems security, network forensics and portable electronic device forensics.

Derrick Kourie is a Professor of Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests are in the area of software engineering.

Cecil Louwrens is an Assistant General Manager with Nedbank and an Adjunct Professor at the University of Johannesburg, Johannesburg, South Africa. His research interests include information security and digital forensics.

David Malan is a doctoral candidate in Computer Science in the Division of Engineering and Applied Sciences at Harvard University, Cambridge, Massachusetts. His research interests include computer security and forensics.

Gavin Manes is a Research Assistant Professor with the Center for Information Security at the University of Tulsa, Tulsa, Oklahoma. His research interests include information assurance, digital forensics, telecommunications security and critical infrastructure protection.

Nasir Memon is a Professor of Computer Science at Polytechnic University, Brooklyn, New York. His research interests include data compression, computer and network security, multimedia data security and multimedia communications.

Robert Mills is an Assistant Professor of Electrical Engineering at the Air Force Institute of Technology, Wright-Patterson AFB, Ohio. His research interests include information security, network management, signal processing and communications systems.

Tyler Moore is a Ph.D. student at the University of Cambridge, Cambridge, United Kingdom. His research interests include telecommunications security, network security analysis, critical infrastructure protection and security economics.

Judie Mulholland is the Research Coordinator at the Florida Cybersecurity Institute, Tallahassee, Florida. Her research interests include cyber crime, trusted computing, digital rights management, data mining and digital forensics.

James Okolica is a First Lieutenant in the U.S. Air Force stationed at the Air Force Institute of Technology, Wright-Patterson AFB, Ohio, where he is pursuing his M.S. degree in Computer Science. His research interests include digital forensics, machine learning and pattern recognition.

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

Mauricio Papa is an Assistant Professor of Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include distributed systems, information assurance, and network and SCADA systems security.

Anthony Persaud is an M.S. student in Computer Engineering at Iowa State University, Ames, Iowa. His research interests include network security and digital forensics.

Gilbert Peterson is an Assistant Professor of Electrical and Computer Engineering at the Air Force Institute of Technology, Wright-Patterson AFB, Ohio. His research interests include digital forensics, image processing, robotics and machine learning.

Cecile Pham was a consultant at Basis Technology Corp., Cambridge, Massachusetts, which develops software for extracting meaningful intelligence from unstructured text.

Scott Piper received his undergraduate degree in Computer Science from the University of Tulsa, Tulsa, Oklahoma. His research interests include digital forensics, network security and software vulnerability analysis.

Mark Pollitt, who is retired from the Federal Bureau of Investigation, is President of Digital Evidence Professional Services, Ellicott City, Maryland. His research interests include digital forensics, enterprise architecture, technology and knowledge management.

Kristina Pyles is an undergraduate student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. Her research interests include computer and portable electronic device forensics, and network security.

Indrajit Ray, Chair, IFIP Working Group 11.9 on Digital Forensics, is an Assistant Professor of Computer Science at Colorado State University, Fort Collins, Colorado. His research interests are in the areas of computer security and digital forensics.

Colette Reekie is a D.Com. student in Informatics at the University of Johannesburg, Johannesburg, South Africa. Her research interests include information security and digital forensics governance issues.

Golden Richard III is an Associate Professor of Computer Science at the University of New Orleans, New Orleans, Louisiana, and the co-founder of Digital Forensics Solutions, LLC. His research interests include digital forensics, mobile computing and operating systems internals.

Vassil Roussev is an Assistant Professor of Computer Science at the University of New Orleans, New Orleans, Louisiana. His research interests include digital forensics, high-performance computing, distributed collaboration and software engineering.

Shelly Seier received her M.S. degree in Computer Science from the University of Tulsa, Tulsa, Oklahoma. Her research interests include digital forensics and network security.

Husrev Sencar is a postdoctoral researcher in the Department of Computer and Information Sciences at Polytechnic University, Brooklyn, New York. His research interests are in the fields of digital forensics and security with an emphasis on multimedia, networking and communications applications.

Sujeet Shenoi is the F.P. Walter Professor of Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include information assurance, digital forensics, critical infrastructure protection and intelligent control.

Micah Sherr is a doctoral candidate in Computer and Information Science at the University of Pennsylvania, Philadelphia, Pennsylvania. His research interests include network security, protocol design and analysis, network intrusion detection and prevention, and privacy.

Marco Slaviero is a Ph.D. candidate in Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include digital forensics and computer security.

Eugene Spafford is a Professor of Computer Sciences, Electrical and Computer Engineering, Philosophy and Communication (courtesy), and the Executive Director of the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University, West Lafayette, Indiana. His research interests include information security and cyber crime issues.

Christopher Stevens is an undergraduate student at Harvard University, Cambridge, Massachusetts.

Kenneth Straka is at IT Security Specialist with the Office of the Inspector General at the National Science Foundation, Arlington, Virginia. His research interests include information assurance, network security and forensics.

Tinus Strauss is a Ph.D. student and a Lecturer in Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include computer networks, digital forensics and finite automata.

Amanda Strickler is an undergraduate student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. Her research interests include digital forensics, cryptography and network security.

Christopher Swenson is a Ph.D. student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include cryptanalysis, network security and digital forensics.

Hein Venter is a Senior Lecturer in Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include digital forensics, privacy, network security, vulnerability analysis and intrusion detection.

Jacobus Venter is a Senior Researcher at the Council for Scientific and Industrial Research, Pretoria, South Africa. His research interests include digital forensics training, evidence mining and probabilistic methods.

Sebastiaan von Solms is a Professor and Head of the Academy for Information Technology at the University of Johannesburg, Johannesburg, South Africa. He is the Vice President of IFIP and the immediate past Chairman of IFIP Technical Committee 11 (Information Security). His research interests are in the area of information security.

Anthony Whitley is an attorney who is retired from the U.S. Department of Justice and the Internal Revenue Service. His research interests include digital forensics, data mining and the application of technology to civil and criminal litigation.

Duminda Wijesekera is an Associate Professor of Information and Software Engineering at George Mason University, Fairfax, Virginia. His research interests are in information security.

Marika Wojcik is an M.Sc. student in Computer Science at the University of Pretoria, Pretoria, South Africa. Her research interests include trust models and modeling trust in computerized environments.

Preface

In the early 1900s, Edmond Locard, the celebrated French criminologist, postulated the Exchange Principle: When any two objects come into contact, both leave traces on each other. Remarkably, this principle holds in the “Brave New World” of silicon chips and the Internet. Even electrons may leave indelible traces.

Digital forensics encompasses the scientific processes for acquiring, preserving, examining, analyzing and presenting electronic evidence. Acquiring and analyzing evidence is challenging: it may involve extracting data from cell phone fragments as happened after the London Underground bombings, capturing child pornography from volatile computer memory, or searching for a single incriminating transaction in four terabytes of data as in a recent FBI investigation. Practically every crime now involves some aspect of electronic evidence; digital forensics provides the techniques and tools to articulate this evidence. Digital forensics also has myriad intelligence applications. And its vital role in information assurance cannot be overlooked—investigations of security incidents yield information that can be used to design more secure systems.

This book, *Advances in Digital Forensics II*, is the second volume in the annual series produced by the 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 the emerging discipline of 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-eight edited papers from the Second Annual IFIP WG 11.9 Conference on Digital Forensics, held at the National Center for Forensic Science, Orlando, Florida, January 29 – February 1, 2006. The papers were selected from fifty submissions, which were refereed by members of IFIP Working Group 11.9 and other internationally-recognized experts in digital forensics.

The chapters are organized into seven sections: themes and issues in digital forensics, evidence collection and handling, forensic techniques, operating system and file system forensics, network forensics, portable electronic device forensics, and training, governance and legal issues. 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 Philip Craiger, Anita Presley and Christopher Swenson for their tireless work on behalf of IFIP Working Group 11.9. We also acknowledge the support provided by the National Center for Forensic Science, Orlando, Florida, the Federal Bureau of Investigation and the National Security Agency.

MARTIN S. OLIVIER AND SUJEET SHENOI

Chapter 1

SOME CHALLENGES IN DIGITAL FORENSICS

Eugene Spafford

Abstract This essay discusses some of the principal challenges facing the emerging discipline of digital forensics. Most of the challenges have a scientific basis—understanding the needs and limitations caused by changes in the scope and pace of information technology. Others are engineering in nature, requiring the construction of new software and hardware to enable the collection, retention and examination of potential digital evidence. All of the challenges have administrative and legal frameworks within which they must be addressed, and the limits and structures imposed by these frameworks must evolve and be shaped by science, engineering and practice.

Keywords: Digital forensics, research challenges, science, engineering, practice

The use of information technology continues to grow at a rapid pace. Sometime in 2005, the world population using the Internet exceeded one billion [6]; estimates are that it will double in less than a decade. Computing devices are being used to communicate, bank, shop, operate businesses, interact with governments, learn and seek entertainment. Simultaneously—and not unexpectedly—criminal activity has risen along with the increase in the user population. One estimate puts global losses from cyber crime at more than \$105 billion per year [2]. Waste and abuse may well match or exceed these figures.

Misuse of information technology resources is a major problem that cannot be addressed solely by better security technologies. The situation is complicated by the need for backwards compatibility, lack of user awareness and education, limits to known technologies, and the massive base of installed infrastructure with little or no support for security. The prospects are dim for near-term solutions to many existing problems [7].

Please use the following format when citing this chapter:

Spafford, E., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 1–9.

If we cannot reengineer our information infrastructure to be completely protected, then we need to address the problems of cyber crime and abuse after they occur: by investigation and corrective action, including application of remedial measures, as well as legal and administrative sanctions. This requires comprehensive tools and technologies for investigation that can be trusted to provide accurate, precise results. It also requires competent investigators who are trained in these tools and technologies so as to draw the correct conclusions.

Digital forensics is a relatively new field. Until the mid 1990s, the only public instances of code and log analysis involved detecting intrusions and misuse, or perhaps making some incidental observations about a potential online miscreant, such as a malware author [5, 8]. In 1992, Cornell juniors David Blumenthal and Mark Pilgrim were arrested for writing and releasing the MBDF virus that targeted Macintosh computers [3]. They were identified by a group of anti-virus researchers (including me) after examining the virus source code and system logs. That same year, the first formal paper about software forensics was written and presented [9]. Thus, we might identify 1992 as the year when digital forensics began to emerge as an identifiable field. However, it is still in a nascent phase where we are trying to identify the scope of the field and to decide what to call it: computer forensics, cyber forensics, digital forensics, digital investigations and so on.

Whatever we call this field, it is, nevertheless, useful to examine its scope and current status. One promising classification is to consider the continuum of science, engineering and practice. Each has its own set of unique challenges and needs. There are no clear demarcations between the three areas, but all three are important and necessary components that need to interconnect and communicate.

- **Science:** We can think of science as the formal investigation and documentation of principles, limitations and structure of a field. Science is performed by formulating hypotheses that can be confirmed or refuted, and then conducting carefully designed experiments or analyses. The outcomes must be meticulously documented and then presented to the community, so others can recreate the results and build on them.

The science of digital forensics is still quite limited. We have seen only a few formal models of how digital forensic investigations are structured and conducted (see, e.g., [4]). We still need to understand the limitations and capabilities of forensic investigations. We also need to examine how the potential for digital forensics can be expanded by designing specialized forensic support within new systems (see, e.g., [1]).

- **Engineering:** Engineering can be viewed as the development and application of tools and procedures to solve real problems with known parameters. For example, engineering addresses the questions of how to reliably find data on known (and new) media, and how to reliably distinguish between multiple alternatives in the recreation of incidents. This has been an area where there has been significant development, although not all of it has been formally conducted and published, especially efforts that have been undertaken by commercial entities.
- **Practice:** The practice of digital forensics does not involve the creation of tools or research into technical issues. Instead, it involves training and the application of known tools and techniques within established limits to address real needs. Practice is a major component of digital forensics and it requires better methods for training and setting standards.

The development of digital forensics has mirrored that of many other disciplines. Problems that arise are initially addressed using tools and techniques developed to solve different but similar problems. Some of the first tools applied to digital forensics were developed for system administration and software debugging. As needs grew, engineering expertise was brought to bear to develop new tools and techniques, leading to specialized procedures and training. The synergy between engineering and practice continues, but new challenges and the anticipation of future needs are now drawing more scientific efforts.

In my keynote lecture at the 2001 Digital Forensics Research Workshop I outlined some of the research challenges I saw in the field of digital forensics. Most of these challenges remain and some have expanded; I do not believe that any of them have been adequately addressed as yet.

New challenges are also presenting themselves as technologies advance. Many of these are related to the incredible growth of storage capacity and the pace of the growth. In 1995, there were about 200 terabytes of storage connected to the Internet; this amount of storage is now accommodated in about 100 commodity computers. However, as reported by IDC and recounted in a 2004 posting to Dave Farber's IP list, worldwide storage connected to the Internet exceeds 30 exabytes—a 150,000-fold increase in capacity in only one decade. This growth in storage is continuing, which means that more information must be protected and more information must be analyzed to discover evidence about incidents.

In the following, I list several open challenges that need to be addressed. Some may be solved by engineering advancements; others require significant scientific investigations.

- How do we copy terabyte (and larger) storage media rather than confiscating them? There is a requirement to obtain and hold evidence, but it is inappropriate to inconvenience innocent victims and third parties who may require the media for business operations. However, the capacity of these media and the available transfer rates to other storage devices make bit-for-bit duplication a major challenge.

Perhaps we must redefine the term “everything needed” when evidence is collected to reduce the amount of data that needs to be copied. Or, we might combine *in situ* examination with copying to reduce the scope of what needs to be copied.

- How do we image large, active disk farms dynamically? Victims and third-party providers may have evidentiary material on their media, but shutting their systems down to make a copy can greatly inconvenience them. Imagine asking Amazon.com or eBay to discontinue service while their drives are being copied!

In addition to the mechanics of copying, we also need to understand how the copies relate to the media at the time of the crime, and we need to present that information according to acceptable standards of evidence.

- Where can storage reside that may contain evidence? In the last few years we have seen an increase in the use of USB “thumb-drives,” cell phones, digital cameras, PDAs, remote storage devices and removable media. Understanding the scope and range of storage continues to be a challenge in forensic investigations. The emerging use of *ad hoc* networks, RAID over network servers, and long-term storage in appliances and home media will blur the notion of “local storage.”

Furthermore, the use of open 802.11 networks in neighborhoods, cable modems and unsecured, unpatched systems means that perpetrators can store data of interest on a number of systems that are not obviously under their control. Not only do we need to be able to examine the evidence on these systems, but we must exclude quickly the actual owners of the systems as suspects.

- How can we trust audit trails? There is always the possibility that an intruder (or his software) may edit or delete the audit trail on a computer, especially a weakly-protected PC. Furthermore, over the last few years we have seen increasingly sophisticated rootkits that dynamically modify the kernels of running systems to hide what is happening—or even to produce false results.

- How do we accurately reconcile evidence collected from multiple machines that may not have accurate clocks to appropriately sequence events? This becomes even more important as more machines are involved, including machines in different time zones.
- How do we deal with non-determinism? A number of systems use randomness or asynchronous events in their processing. If we need to understand how something happened, how do we recreate the relevant events and their timings? How do we accomplish this if the input comes from *ad hoc* networks of sensors that cannot be “reset” to an earlier state?
- How do we cope with the changing nature of what needs to be investigated? Instead of simple data and image files, we are now seeing video, audio, GIS material, VoIP systems, sensor net data, SCADA systems and more. What are the standards for terminology, collection and representation that will allow investigations to be conducted accurately on these systems?
- What are the limits over time of what we can do? If we perform forensic examinations of backups and mirror sites, how much can we accurately conclude?

Several of the challenges have a scientific basis—understanding the needs and limitations caused by changes in the scope and pace of information technology. Others are engineering in nature, requiring the construction of new software and hardware to enable the collection, retention and examination of potential digital evidence. All of these challenges have administrative and legal frameworks within which they need to be addressed, and the limits and structures imposed by these frameworks need to evolve and be shaped by the science of what is possible, by the availability of engineered solutions and by disciplined practice.

In Asimov’s 1956 short story, *The Dead Past*, a scientist helps develop an inexpensive “time viewer” that allows one to see incidents from the past. After much thought, his superiors decide that no one should know about their discovery. The scientist was stunned. He believed that the world would benefit from the technology: old crimes could be viewed and solved, the causes of accidents could be traced, historical disputes could be settled. The scientist acknowledged that for certain incidents, such as those involving religious figures, disclosures from the past might be traumatic, but they would be therapeutic in the long term.

The scientist made copies of the blueprints and sent them to newspapers and leaders around the world. Only after the blueprints went out did his superiors find out what he had done. It was then that they

pointed out that the past is not simply long dead events, but also very recent incidents. By tuning the viewer to half a second in the past, one could snoop on what anyone was doing *right now*. The tragedy was that those who wanted to snoop on others could do so. The ability to see the past resulted in the loss of privacy for all.

This is an apt parable for the digital forensics community. Our ability to analyze data from the past makes it possible to examine the current behavior and activities of those we might wish to monitor. This capability presents opportunities, but also new responsibilities.

Clearly, there are significant ethical issues that must be addressed concomitantly with technological advancements in digital forensics. These issues need to be identified, resolved and articulated. As digital forensic technology improves, it may well become easier to discover details about people's lives, loves and activities, especially as more information is stored online and maintained indefinitely. Digital forensic practitioners have a duty to identify the guilty and exonerate the innocent in issues of misbehavior and crime, but they should also ensure that they preserve the privacy of all parties, principals as well as incidental contacts. Privacy, once violated, is difficult—if not impossible—to restore.

Digital forensics is an engaging, vibrant field. Many challenges exist, but opportunities abound to innovate and make a difference. We ought to refrain from needless arguments about what to call the discipline and look beyond the next set of problems to address. We should consider how we want to be known and what roles we should play. Technical challenges may shape what we do, but come what may, there will always be a role for sound human judgment.

References

- [1] F. Buchholz, Pervasive Binding of Labels to System Processes, Ph.D. Dissertation, CERIAS Technical Report 2005-54, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 2005.
- [2] Cable News Network (CNN), Record bad year for tech security (money.cnn.com/2005/12/29/technology/computer_security/index.htm), December 29, 2005.
- [3] J. Carmona, Computer virus traced to Cornell students, *The Cornell Daily Sun*, February 25, 1992.
- [4] B. Carrier, A Hypothesis-Based Approach to Digital Forensic Investigations, Ph.D. Dissertation, CERIAS Technical Report 2006-06, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 2006.

- [5] T. Longstaff and E. Schultz, Beyond preliminary analysis of the WANK and OILZ worms: A case study of malicious code, *Computers and Security*, vol. 12(1), pp. 61-77, 1993.
- [6] Miniworld Marketing Group, Internet world stats (www.internetworldstats.com/stats.htm), June 1, 2006.
- [7] President's Information Technology Advisory Committee (PITAC), *Cyber Security: A Crisis of Prioritization*, National Coordination Office for Information Technology Research and Development, Arlington, Virginia, 2005.
- [8] E. Spafford, The Internet worm program: An analysis, *Computer Communication Review*, vol. 19(1), pp. 17-57, 1989.
- [9] E. Spafford and S. Weeber, Software forensics: Can we track code to its authors? *Computers and Security*, vol. 12(6), pp. 585-595, 1993.

Chapter 2

ADVANCED FORENSIC FORMAT: AN OPEN EXTENSIBLE FORMAT FOR DISK IMAGING

S. Garfinkel, D. Malan, K. Dubec, C. Stevens and C. Pham

Abstract This paper describes the Advanced Forensic Format (AFF), which is designed as an alternative to current proprietary disk image formats. AFF offers two significant benefits. First, it is more flexible because it allows extensive metadata to be stored with images. Second, AFF images consume less disk space than images in other formats (e.g., Encase images). This paper also describes the Advanced Disk Imager, a new program for acquiring disk images that compares favorably with existing alternatives.

Keywords: Disk imaging, image storage, Advanced Forensic Format (AFF)

1. Introduction

Most forensic practitioners work with just one or a few disks at a time. A wife might bring her spouse's laptop to an examiner to make a copy a few days before she files for divorce. Police might raid a drug dealer's apartment and seize a computer that was used for contacting suppliers. In these cases, it is common practice to copy the drive's contents sector-for-sector into a single file, a so-called "raw" or "dd" copy. Some practitioners make a sector-for-sector copy of the original disk to a second drive that is the same size as the original.

Raw images are widely used because they work with practically every forensic tool available today. However, raw images are not compressed, as a result, they can be very large—even if the drive itself contained very little data.

The obvious way to solve the problem of data storage is to use a file compressor like `gzip` [9] or `bzip2` [20]. But neither `gzip` nor `bzip2` allow

Please use the following format when citing this chapter:

Garfinkel, S., Malan, D., Dubec, K., Stevens, C., Pham, C., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 13–27.

random access within a compressed file. Because many forensic tools require random access to captured data just like a file system requires random access to a physical disk, disk images that are compressed must be decompressed before they can be used.

A second problem with raw images is the storage of data about the image itself, i.e., its metadata. Because a raw image is a sector-for-sector copy of the drive under investigation, the file cannot store metadata such as the drive's serial number, the name of the investigator who performed the acquisition, or the date on which the disk was imaged. When metadata is not stored in the image file itself, there is a chance that it will become separated from the image file and lost—or even confused with the metadata of another drive.

After evaluating several of the existing alternatives for storing disk images, we decided to create the new Advanced Forensic Format (AFF™) for our forensic work. AFF is open and extensible, and unencumbered by patents and trade secrets. Its open-source implementation is distributed under a license that allows its code to be freely integrated into other open-source and propriety programs.

AFF is extensible—new features can be added in a manner that maintains forward and backward compatibility. This extensibility allows older programs to read AFF files created by newer programs, and it allows newer AFF programs to read older AFF files that lack newer features.

This paper presents our research on the design and implementation of AFF. We have used AFF to store more than one terabyte of data from imaged hard drives using less than 200 GB of storage. We are currently working to improve the functionality and performance of the AFF implementation and associated tools.

2. Related Work

Several formats exist for forensic images, but none offers AFF's combination of openness and extensibility. This section surveys the most common formats in use today. Some commercially-available tools support formats other than their own. Nevertheless, support for proprietary formats, which is often the result of reverse engineering, tends to be incomplete.

2.1 EnCase Format

Guidance Software's EnCase Forensic [12] is perhaps the *de facto* standard for forensic analysis in law enforcement. It uses a proprietary format for images based on ASR Data's Expert Witness Compression Format [4]. EnCase's Evidence File format (Figure 1) contains a physical

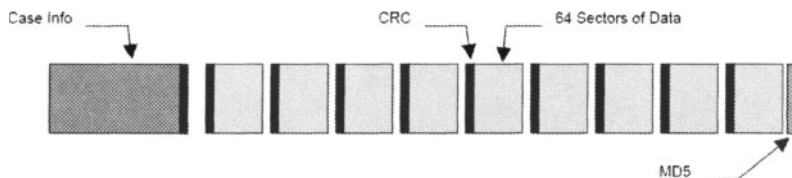


Figure 1. EnCase format.

bitstream of an acquired disk, prefixed with a “Case Info” header, interlaced with CRCs for every block of 64 sectors (32 KB), and followed by a footer containing an MD5 hash for the entire bitstream. The header contains the date and time of acquisition, examiner’s name, notes on the acquisition, and an optional password; the header concludes with its own CRC.

The EnCase format is compressible and searchable. Compression is block-based [17], and “jump tables” and “file pointers” are maintained in the format’s header or between blocks “to enhance speed” [14]. Disk images can be split into multiple files (e.g., for archival to CD or DVD).

A limitation of the EnCase format is that image files must be less than 2 GB in size. As a result, EnCase images are typically stored in directories with the individual file’s given names (e.g., FILE.E01, FILE.E02, etc.). The format also limits the type and quantity of metadata that can be associated with an image. Some vendors have achieved limited compatibility by reverse-engineering the format; however, these attempts are generally incomplete.

Table 1. Comparison of AFF and EnCase (all values in MB).

		Zeroes	Shakespeare	Random
AFF	-X1	28	2879	6301
	-X6	6	2450	6301
	-X9	6	2443	6301
Encase	“Good”	33	3066	6303
	“Best”	12	2846	6303

Table 1 compares the sizes of AFF and EnCase images of a 6 GB hard drive. The hard drive was filled with: (i) all zeroes, (ii) the complete works of William Shakespeare repeated approximately 1,200 times, and (iii) random data. AFF used **gzip** compression performed at levels “1,” “6” and “9” with **aimage** -X1, -X6 and -X9 options. EnCase compression was performed using the “Good” and “Best” levels.

2.2 Forensic Toolkit (FTK) Formats

AccessData's Forensic Toolkit (FTK) [1] is a popular alternative to EnCase. It supports the storage of disk images in EnCase's file format or SMART's file format (Section 2.9), as well as in raw format and an older version of Safeback's format (Section 2.7).

2.3 ILook Formats

ILook Investigator v8 [15] and its disk-imaging counterpart, IXimager, offer three proprietary, authenticated image formats: compressed (IDIF), non-compressed (IRBF) and encrypted (IEIF). Few technical details have been disclosed publicly. However, IXimager's online documentation [15] provides some insights: IDIF "includes protective mechanisms to detect changes from the source image entity to the output form" and supports "logging of user actions within the confines of that event." IRBF is similar to IDIF, except that disk images are left uncompressed. IEIF encrypts disk images. To facilitate compatibility with ILook Investigator v7 and other forensic tools, IXimager allows for the transformation of each of these formats into raw format.

2.4 ProDiscover Format

Technology Pathways' ProDiscover family of security tools [23] uses the ProDiscover Image File Format [22]. It consists of five parts: a 16-byte image file header, which includes a signature and version number for an image; a 681-byte image data header, which contains user-provided metadata about the image; image data, which comprises a single block of uncompressed data or an array of blocks of compressed data; an array of compressed blocks sizes (if the image data is compressed); and i/o log errors describing any problems during the image's acquisition. The format is fairly well documented, but it is not extensible.

2.5 PyFlag Format

PyFlag [17] is a "Forensic and Log Analysis GUI" developed by the Australian Department of Defence. It uses **sgzip**, a seekable variant of the **gzip** format. (PyFlag can also read and write ASR Data's Expert Witness Compression Format [19].) By compressing blocks (32 KB by default) individually, **sgzip** allows for rapid accessing of a disk image by forensic software without the need to first decompress the entire image. The format does not associate metadata with images [18, 19].

2.6 RAID Format

Relatively few technical details of DIBS USA's Rapid Action Imaging Device (RAID) [8] are publicly available. It offers "built-in integrity checking" and is designed to create an identical copy in raw format of one disk on another. The copy can then "be inserted into a forensic workstation" [7].

2.7 SafeBack Format

SafeBack [3] is a DOS-based utility designed to create exact copies of entire disks or partitions. It offers a "self-authenticating" format for images, whereby SHA-256 hashes are stored along with data to ensure the integrity of images. SafeBack's developers claim that the software "safeguards the internally stored SHA-256 values" [3].

2.8 SDi32 Format

Vogon International's SDi32 [25] imaging software is designed to be used with write-blocking hardware. It is capable of making identical copies of disks to tape, disk or file, with optional CRC32 and MD5 fingerprints. The copies are stored in raw format.

2.9 SMART Formats

SMART [5] is a software utility for Linux designed by the original authors of Expert Witness (now sold under the name EnCase [12]). It can store disk images as pure bitstreams (compressed or uncompressed) or in ASR Data's Expert Witness Compression Format [4]. Images in the latter format can be stored as a single file or in multiple segment files, consisting of a standard 13-byte header followed by a series of sections, each of type "header," "volume," "table," "next" or "done." Each section includes its type string, a 64-bit offset to the next section, its 64-bit size, padding, and a CRC, in addition to actual data or comments, if applicable. Although the format's "header" section supports free-form notes, an image can have only one such section (in its first segment file only).

2.10 Comparison of Formats

Table 1 provides a comparison of the features offered by various file formats. A format is considered to be "non-proprietary" if its specification is publicly available. It is "extensible" if it supports the storage of arbitrary metadata. It is "seekably compressed" if it can be searched without being uncompressed in its entirety. A bullet (•) indicates sup-

Table 2. Summary of features supported by various file formats.

	Extensible	Non-Proprietary	Compressed & Seekable
AFF	•	•	•
EnCase			•
ILook	?		•
ProDiscover		•	•
PyFlag		•	•
RAID		•	
SafeBack	?		?
SDi32		•	
SMART		•	

port for a feature, while a question mark (?) indicates that support for a feature is not disclosed publicly. FTK is omitted because it uses other tools' formats.

2.11 Digital Evidence Bags

Turner proposed the concept of a Digital Evidence Bag (DEB) [24] as a “wrapper” or metaformat for storing digital evidence from disparate sources. The DEB format consists of a directory that includes a tag file, one or more index files, and one or more bag files. The tag file is a text file that contains metadata such as the name and organization of the forensic examiner, hashes for the contained information, and data definitions. Several prototype tools have been created for DEBs, including a bag viewer and a selective imager.

3. Advanced Forensic Format

The Advanced Forensic Format (AFF) is a single, flexible format that can be used for a variety of tasks. This section discusses AFF's design goals and its two-layered architecture that helps achieve the design goals.

3.1 AFF Goals

The specific design goals for AFF are provided below. We believe that AFF delivers on all these goals.

- Ability to store disk images with or without compression.
- Ability to store disk images of any size.
- Ability to store metadata within disk images or separately.

- Ability to store images in a single file of any size or split among multiple files.
- Arbitrary metadata as user-defined name/value pairs.
- Extensibility.
- Simple design.
- Multiple platform, open source implementation.
- Freedom from intellectual property restrictions.
- Provisions for internal self-consistency checking, so that part of an image can be recovered even if other parts are corrupted or otherwise lost.
- Provisions for certifying the authenticity of evidence files with traditional hash functions (e.g., MD5 and SHA-1) and advanced digital signatures based on X.509(v)3 certificates.

3.2 AFF Layers

Many of the design goals are accomplished by partitioning the AFF format into two layers: the disk-representation layer and the data-storage layer. The disk-representation layer defines a schema that is used for storing disk images and associated metadata. The data-storage layer specifies how the named AFF segments are stored in an actual file. We have developed two data-storage implementations.

3.2.1 AFF Disk-Representation Layer. AFF's disk-representation layer defines specific segment names that are used for representing all the information associated with a disk image. Each AFF segment consists of a segment name, a 32-bit "flag," and a data payload. The name and the data payload can be between 0 and $2^{32} - 1 = 4,294,967,295$ bytes long. In practice, segment names are less than 32 bytes, while the data payload is less than 16 MB.

AFF 1.0 supports two kinds of segments: metadata segments, which are used for holding information about the disk image, and data segments called "pages," which are used for holding the imaged disk information itself.

Metadata segments can be created when a disk is accessioned or after a disk is imaged. For example, `device.sn` is the name of the segment used to hold the disk's serial number, while `date.acquired` holds the time that the disk image was acquired.

Two special segments are **accession_gid**, which holds a 128-bit globally unique identifier that is different each time the disk imaging program is run, and **badflag**, which holds a 512-byte block of data that identifies blocks in the data segments that are “bad” (i.e., cannot be read). The existence of the **badflag** makes it possible for forensic tools to distinguish between sectors that cannot be read and sectors that are filled with NULLs or another form of constant data, something that is not possible with traditional disk forensic tools. Tools that do not support the **badflag** will interpret each “bad” sector as a sector that begins with the words “BAD SECTOR” followed by random data; these sectors can thus be identified as being bad if they are encountered by a human examiner. Alternatively, AFF can be configured to return bad sectors as sectors filled with NULLs. Table 3 provides a complete list of the segment names defined in AFF 1.0.

Data segments hold the actual data copied from the disk being examined. All data segments must be the same size; this size is determined when the image file is created and stored in a segment named **pagesize**. Although **pagesize** can be any value, common choices are 2^{20} (1 MB) and 2^{24} (16 MB).

Segments are given sequential names **page 0**, **page 1**, ..., **page n**, where **n** is as large as is necessary. The segment (page) number and byte offset within the uncompressed segment of any 512-byte sector i in the AFF file are determined by the formulas:

$$\text{page number} = \frac{i \times 512}{\text{pagesize}} \quad (1)$$

$$\text{offset} = (i \times 512) - (\text{page \#}) \times \text{pagesize} \quad (2)$$

AFF data pages can be compressed with the open-source **zlib** [10] or they can be left uncompressed. The data page’s 32-bit flag encodes if the page is compressed or not. Compressed AFF files consume less space but take more time to create; this is because the time taken to write uncompressed data is typically shorter than the combined time taken to compress data and write the compressed data. Interestingly, compressed files can actually be faster to read than uncompressed files as modern computers can read and uncompress data faster than they can read the equivalent uncompressed data of a hard drive. The decision to compress or not to compress can be made when an image is acquired. Alternatively, an uncompressed file can be compressed later.

Checksums and signatures can be used to certify that data in the image data segments has not been accidentally or intentionally modified

Table 3. AFF segments defined in the AFF 1.0 specification.

Segment Name	AFFLIB Symbol	Meaning
<i>Housekeeping Segments</i>		
—	AF_IGNORE	Ignore this segment; zero-length name.
dir	AF_DIRECTORY	AFF directory; should be the last segment.
<i>AFF Segments (pertaining to the entire image)</i>		
pagesize	AF_SEGSIZE	Size (in bytes) of each uncompressed AFF data page is stored in segment "flag" field.
imagesize	AF_IMAGESIZE	Size (in bytes) of the complete image is stored in the segment data area. (8-byte value).
badsectors	AF_BADSECTORS	Number of 512-byte sectors in the image that were marked as "bad."
badflag	AF_BADFLAG	512-byte flag that is used to denote sectors in the image file that could not be read from the media.
blanksectors	AF_BLANKSECTORS	Number of sectors in the image file that are completely blank (i.e., filled with 512 ASCII NULLs.)
<i>AFF Pages (repeated for each data segment)</i>		
page%d	AF_PAGE	The actual data of page %d. Data is compressed if the AF_PAGE.COMPRESSED bit in the flag is set.
page.md5.%d	AF_PAGE_MD5HASH	MD5 of the uncompressed page.
page.md5.sig%d	AF_PAGE_MD5SIG	PKCS #7 signature of page %d's MD5.
page.sha1.%d	AF_PAGE_SHA1HASH	SHA-1 of the uncompressed page.
page.sha1.sig%d	AF_PAGE_SHA1SIG	PKCS #7 signature of page %d's SHA-1.
<i>Some AFF Segments created by the Advanced Disk Imager</i>		
case_num	AF_CASE_NUM	Case number for EnCase compatibility.
image_gid	AF_IMAGE_GID	A unique 128-bit image identifier.
imaging_commandline	AF_IMAGING_COMMAND_LINE	Complete command used to create image.
imaging_date	AF_IMAGING_DATE	Date and time when imaging was started.
imaging_notes	AF_IMAGING_NOTES	Notes made by the forensic examiner when imaging was started.
imaging_device	AF_IMAGING_DEVICE	Device used as the source of the image.

after it was acquired. AFF 1.0 supports two modes of certification: hashes and digital signatures.

It is common practice in the forensic community to record MD5 or SHA-1 hashes of an image. AFF 1.0 allows these hashes to be recorded for the entire image and for each individual data segment. The hash values are stored in specially named segments.

AFF 1.0 provides an additional level of certification by supporting the inclusion of PKCS #7 [16] digital signatures. These signatures can also be recorded for the entire disk image and for individual data segments. Because signatures are calculated on uncompressed data, it is possible to acquire and digitally sign a disk image and then compress the image without compromising the integrity of the digital signatures.

3.2.2 AFF Data-Storage Layer. The data-storage layer handles the task of storing the actual AFF segments. AFF 1.0 has two storage formats. The AFF 1.0 Binary Format specifies that segments are stored sequentially in one or more files. AFF 1.0 XML, on the other hand, stores information in the XML format. The binary files are smaller and faster than the XML files. However, the standardized format of the XML files makes them easier to use with non-forensic tools. In practice, it is possible to store the actual disk image in an AFF binary file and the disk's metadata in an XML file, although this does introduce the possibility that the two files might become separated.

The original plan for AFF's Binary Format was to use an open-source B-tree implementation like Berkeley DB (BDB), the GNU Database Manager (GDBM), or a similar system to store the segments as a series of name/value pairs. But BDB and GDBM are distributed under the GNU Public License, which is unacceptable to many commercial software developers. (A licensed version of BDB with fewer restrictions can be obtained from SleepyCat Software [21], but not for free.) The Apache SDBM database has no such restrictions. However, our tests indicate that SDBM creates sparse files exceeding 100 GB when used to store just a few GB of image data. Note that because the files are sparse, they do not take up all the space on disk. However, if a file is copied to another disk, the unmapped sectors are filled with zeroes. Another concern is that the layout of information in a B-tree file might be too complex to explain in court, should the need arise.

Consequently, we adopted a simple approach for storing AFF segments in a binary file. Each AFF 1.0 binary file consists of a header followed by zero or more binary AFF segments. Each binary AFF segment consists of a header, a variable-length segment name, a 32-bit flag, a variable-length data area and a segment footer. The segment's length

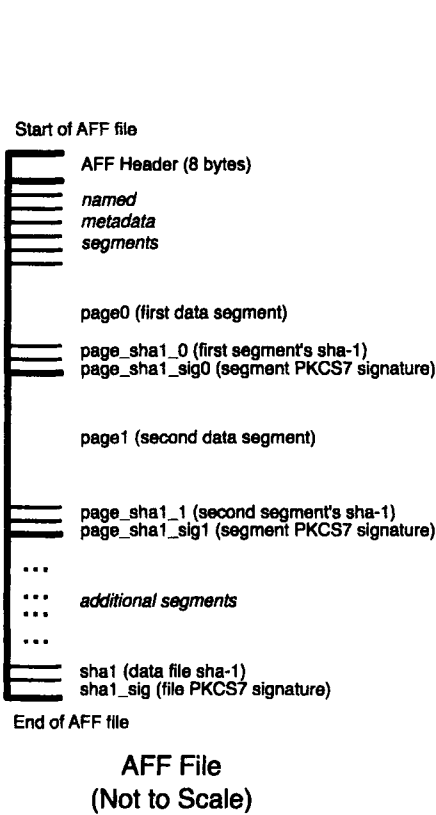


Figure 2. AFF file schematic.

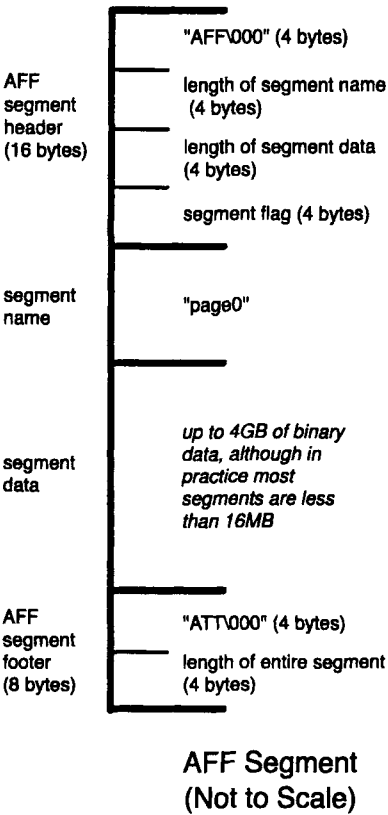


Figure 3. AFF segment schematic.

is stored in the header and footer, allowing for rapid bidirectional seeking as only the headers and the footers need to be read.

Figures 2 and 3 present schematic representations of an AFF file and segment, respectively. The AFF file schematic shows the file's header and AFF segments. The AFF segment schematic shows the segment's header, name, data and segment tail. Including the segment's length at the beginning and the end makes it possible to rapidly seek through a large AFF file without having to read all of the data in each segment.

The AFF XML 1.0 Format is a simplistic encoding of name/value segments as a series of XML objects. This XML encoding can be produced automatically from an AFF binary file using the `afxml` tool that we have developed.

4. AFF Library and Tools

AFFLIB™ is an implementation of AFF 1.0 written in C/C++. To avoid forcing the programmer to understand segments, data segments, compression, etc., AFFLIB implements a simple abstraction that makes the AFF image file appear as two resources: a simple name/value database that can be accessed with traditional `put` and `get` semantics and a stream that can be accessed using `af_open()`, `af_read()` and `af_seek()` function calls. If `af_open()` is used to open a non-AFF file, the library defaults to a pass-through mode of operation, allowing AFF-aware programs to work equally well with raw files.

AFFLIB also supports an `af_write` function call—even across compressed data segments—which makes it easier to write disk imaging programs.

The AFF source code comes with a set of tools including AImage: Advanced Disk Imager (`aimage`), a program for converting AFF metadata into XML (`afxml`), a program for converting raw images to AFF images and back (`afconvert`).

In addition, we have modified Sleuth Kit [6] to work with AFF files. The modifications are relatively minor and have already been incorporated into Sleuth Kit's distribution. When run interactively, no performance degradation is noticeable—not even on compressed AFF files.

5. Advanced Disk Imager

In the early days of our forensics work, we acquired disk images using the Unix `dd` [2] program. Although running `dd` with the correct arguments can be very fast, the program has several shortcomings. The `dd` program does not automatically retry read attempts on bad sectors. If there is a bad region on the disk, `dd` attempts to plod its way through the region, which can take a very long time. It is easy to corrupt a disk by accidentally reversing or otherwise mistyping the values of the `if=` and `of=` arguments. Finally, `dd` does not capture forensic metadata such as the serial number of a hard drive or even the time of the acquisition.

Consequently, we decided to create a new disk imaging program. Our Advanced Disk Imager (`aimage`) was developed specifically for the demanding task of imaging large numbers of hard drives purchased on the secondary market. We have tested `aimage` on several hundred different hard drives and have found it to be a very powerful disk-imaging tool.

Written in C++, `aimage` is designed to be run from the command line. It can image any device that is visible to the operating system. The program can also create an image from data sent to it over a network using an unencrypted or encrypted TCP connection.

```

Terminal — ssh — 80x24
IMAGING Thu Nov 10 10:53:27 2005
Source device: /dev/ad2 AFF Output: /project/junk.aff
Model #: QUANTUM FIREBALL ST3.2A
firmware: A8F.0000 Sector size: 512 bytes
S/N: 153718340531 Total sectors: 6,306,048

-----
Currently reading sector: 97,792 (512 sectors at once)
Sectors read: 98,304 ( 1.56%) # blank: 1,026

Time spent reading: 00:00:05 Estimated total time left: 00:21:34
Total bytes read: 50,331,648

Compressed bytes written: 25,735,396
Time spent compressing: 00:00:09
Overall compression ratio: 48.87% (0% is none; 100% is perfect)
Free space on 192.168.1.1:/project: 68,937 MB (12.44%)

```

Figure 4. Screenshot of **aimage** in action.

The **aimage** program can create a raw file (in the so-called “dd” format), an AFF file, or both files at the same time. The program offers a real-time display on a 80×24 screen detailing the status of the image capture. By using a text-based interface, **aimage** can be run from a bootable Linux distribution on computers that do not support the X Window System. A screenshot of **aimage** is shown in Figure 4.

A sophisticated error-handling system causes **aimage** to read large data blocks unless it encounters an error, in which case it switches to a smaller block size and attempts to re-read the data. If it encounters too many errors in a row the program goes to the end of the disk and attempts to image the remaining disk sectors in reverse. This approach, pioneered by the **dd_rescue** [11] forensics program, works well to recover disks that have a single region of bad blocks. A future version of **aimage** will allow the remaining bad region to be further bisected so that additional good blocks can be recovered.

The **aimage** program calculates MD5 and SHA-1 hashes of a disk as the disk is imaged. Plans are underway to implement digital signatures as prescribed by the AFF standard.

6. Conclusions

The Advanced Forensic Format (AFF) is an attractive, tested system for storing forensic disk images. With the Advanced Disk Imager we

have collected images from nearly a thousand hard drives over the past year. AFF's tools have enabled us to work with these images quickly, efficiently and at a very high level. AFFLIB's support of additional file formats, e.g., EnCase and Expert Witness, permits the tools to be used for a variety of disk image formats without the need for modification or translation. AFF and AFFLIB may be downloaded from www.afflib.org.

Acknowledgements

Brian Carrier and Peter Wayner provided useful feedback on the initial design of the AFF system. Basis Technology Corp. provided substantial funding for the AFF effort. Simson Garfinkel was supported by a fellowship from the Center for Research on Computation and Society, Division of Engineering and Applied Sciences, Harvard University. David Malan was partially supported by NSF Trusted Computing Grant CCR-0310877. The support of this work by Michael D. Smith of Harvard University is greatly appreciated. AFF and AFFLIB are trademarks of Simson Garfinkel and Basis Technology, Inc.

References

- [1] AccessData, Forensic Toolkit (www.accessdata.com/products/ftk).
- [2] Apple Developer Connection, dd, BSD General Commands Manual (developer.apple.com/documentation/Darwin/Reference/Man/pages/man1/dd.1.html).
- [3] Armor Forensics, SafeBack (www.forensics-intl.com/safeback.htm).
- [4] ASR Data Acquisition and Analysis, Expert Witness Compression Format Specification (www.asrdata.com/SMART/whitepaper.html), April 7, 2002.
- [5] ASR Data Acquisition and Analysis, SMART (www.asrdata.com/SMART).
- [6] B. Carrier, The Sleuth Kit & Autopsy: Forensic Tools for Linux and other Unixes (www.sleuthkit.org), 2005.
- [7] DIBS USA, Computer Forensics (www.dibsusa.com).
- [8] DIBS USA, DIBS RAID – Rapid Action Imaging Device (www.dibsusa.com/products/raid.html).
- [9] J. Gailly and M. Adler, The gzip Home Page (www.gzip.org), 2003.
- [10] J. Gailly and M. Adler, zlib (v.1.2.3) (www.zlib.net), 2005.
- [11] K. Garloff, dd_rescue (www.garloff.de/kurt/linux/ddrescue), August 28, 2004.

- [12] Guidance Software, EnCase Forensic (www.guidancesoftware.com/products/ef.index.asp).
- [13] Guidance Software, EnCase Forensic Edition User Manual, Version 4 (www.guidancesoftware.com/support/downloads.asp).
- [14] Guidance Software, *EnCase Legal Journal*, April 2004.
- [15] Internal Revenue Service, ILook v8 – Computer Forensic Application, IRS Criminal Investigation Division – Electronic Crimes, Washington, DC (www.ilook-forensics.org/homepage.html).
- [16] B. Kaliski and K. Kingdon, Extensions and Revisions to PKCS #7 (ftp.rsasecurity.com/pub/pkcs/pkcs-7/pkcs-7v16.pdf), 1997.
- [17] PyFlag, Advanced Open Standard Forensics Format (pyflag.sourceforge.net/Documentation/articles/forensic.format.html).
- [18] PyFlag, Disk Forensics (pyflag.sourceforge.net/Documentation/tutorials/forensics.html).
- [19] PyFlag, PyFlag IO Sources (pyflag.sourceforge.net/Documentation/manual/iosource.html).
- [20] J. Seward, bzip2 and libbzip2 (www.bzip.org/index.html).
- [21] Sleepycat Software (www.sleepycat.com).
- [22] Technology Pathways, ProDiscover Image File Format (v.1.3) (www.techpathways.com/uploads/ProDiscoverImageFileFormatv4.pdf).
- [23] Technology Pathways, The ProDiscover Family of Computer Security Tools (www.techpathways.com/DesktopDefault.aspx?tabindex=3&tabid=12).
- [24] P. Turner, Unification of digital evidence from disparate sources (digital evidence bags), *Proceedings of the Fifth Annual Digital Forensics Research Workshop*, 2005.
- [25] Vagon International, Imaging Software (www.vagon-forensic-hardware.com/forensic-hardware/data-capture/advanced-imaging-software.htm).

Chapter 3

FILE SYSTEM SUPPORT FOR DIGITAL EVIDENCE BAGS

Golden Richard III and Vassil Roussev

Abstract Digital Evidence Bags (DEBs) are a mechanism for bundling digital evidence, associated metadata and audit logs into a single structure. DEB-compliant applications can update a DEB's audit log as evidence is introduced into the bag and as data in the bag is processed. This paper investigates native file system support for DEBs, which has a number of benefits over ad hoc modification of digital evidence bags. The paper also describes an API for DEB-enabled applications and methods for providing DEB access to legacy applications through a DEB-aware file system. The paper addresses an urgent need for digital-forensics-aware operating system components that can enhance the consistency, security and performance of investigations.

Keywords: Operating system internals, file systems, digital evidence bags

1. Introduction

Digital forensic tools typically utilize standard operating system components, e.g., file systems and caching mechanisms. However, there are compelling performance, consistency and security reasons for making operating system components “digital forensics aware.” These include performance (e.g., better data distribution and clustering mechanisms, particularly for distributed digital forensics [2]), security (e.g., protection of digital evidence from unauthorized access or tampering), and consistency. This paper considers the advantages and design challenges of digital-forensics-aware file systems. Specifically, it examines how auditing of digital evidence is currently handled and how an enhanced file system can make this process more automated and more accurate.

Evidence bags and seals are a standard item in traditional crime scene investigations. Bags and seals allow evidence to be preserved and catego-

Please use the following format when citing this chapter:

Richard, G. III, Roussev, V., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 29–40.

rized, and tamper-evident designs indicate if the evidence is still secure. Many types of evidence bags provide ample writing space so that notes can be written directly on the bag. Furthermore, the bag's seal may include information such as the name of the investigating officer, case identifiers, the suspect's name, a description of the item, and the date and time when the bag was sealed. Continuity sections on the bag permit tracking the movements of the bag and noting its chain of custody.

Traditional digital forensic methods capture, preserve and analyze evidence in standard electronic containers: images of seized hard drives (e.g., created using the Unix `dd` command) are stored in regular files and documents are typically processed "as is." Auditing a digital investigation, from identification and seizure of evidence through duplication and analysis is essentially ad hoc, with much of the information recorded in separate log files or in the investigator's case notebook.

For example, the popular `dd` utility provides no direct method for capturing information about when the imaging operation took place, who performed the operation, and the results of integrity checks. This information must be recorded separately and, in the case of an integrity check, additional commands (e.g., `md5sum` or a similar cryptographic hashing command) must be executed and the output recorded manually. While enhanced versions of `dd` exist, e.g., `dcfldd`, adding integrity checks to each application is tedious and error-prone. In addition, integrity problems are aggravated when large chunks of digital evidence must be split into pieces, for example, when a disk image is fragmented to fit on removable storage, and then reassembled for processing.

Ad hoc auditing is bound to be incomplete. Because different tools provide widely disparate amounts of auditing information, much of the information must be recorded manually by an investigator. Over the course of an investigation, a piece of digital evidence may be touched by many different tools, some of which generate no audit trail (e.g., `dd` and Sleuth Kit command line tools [5, 4]) and some that generate their own audit logs (e.g., FTK [1]). Ultimately, an investigator is left to piece together these bits of audit trail information to create a comprehensive view of what occurred during the investigation. The failure to record certain information, e.g., the MD5 hash generated by `md5sum` for a large disk image, could result in a huge amount of lost time if the operation must be repeated.

Digital Evidence Bags (DEBs) [8] are universal containers for digital evidence, much as traditional evidence bags are containers for other types of forensic evidence. DEBs bundle digital evidence, associated metadata and audit logs into a single structure, providing an audit trail of operations performed on the evidence as well as integrity checks. In

addition to providing increased security for digital evidence, the audit log details the processes applied to the evidence throughout an investigation. This is potentially useful from the educational and evaluation standpoints, allowing novice investigators to see what steps were taken, which tools were used, and the order in which they were used. DEB-compliant applications can update a DEB's audit log as evidence is introduced into the bag and as data in the bag is processed.

This paper suggests that while the adoption of a standard format for storing digital evidence could radically improve the investigative process, system support for DEBs would be even more useful. The paper investigates native file system support for DEBs in which the basic file type is a DEB. This approach provides several benefits over ad hoc DEB implementations. Some of the advantages of DEBs can be realized even for current DEB-unaware tools: a DEB-enabled file system can transparently offer a DEB's contents to such tools while automatically updating the DEB's metadata and audit log. Another advantage, even for DEB-enabled tools, is that the code for accessing a DEB, including introducing and removing items from the bag and updating the audit log and metadata, needs to be certified only once. Finally, a standard API for accessing DEBs greatly reduces the effort in involved adding DEB support to current and future applications.

2. Basic DEB Structure

The notion of a Digital Evidence Bag (DEB) was proposed by Turner [8]. This section discusses the DEB structure at a high level, focusing instead on the DEB components necessary for native file system support. Interested readers are referred to [8] for additional details about DEBs.

A DEB consists of a collection of objects. The first is the tag area, which is a set of name/value pairs containing metadata associated with the DEB. The information includes a unique identifier for the DEB, the creation date and time, the name and organization of the DEB creator, and a list of Evidence Units (EUs) stored in the DEB. Each EU provides a name for a distinct blob of digital evidence stored in the bag and the blob's associated index file. An index file describes one blob of digital evidence, e.g., detailing the files contained in the blob or the physical characteristics and model/serial number of an imaged disk device. Finally, an audit log (called a Tag Continuity Block in [8]) tracks the operations performed against a DEB, including the date, time, affected blocks, application signature of each operation as well as periodic hashes of DEB contents. Section 3.4 discusses how secure auditing techniques can be used to protect DEB contents from tampering.

3. Design Overview

Native file system support for DEBs must:

- Allow transparent import of DEBs into an enabled file system, i.e., support the automatic conversion of DEBs to a native storage format.
- Allow transparent export of DEBs from an enabled file system to a “normal” file system, i.e., support the automatic conversion of DEBs to a popular container format, e.g., XML.
- Provide convenient, efficient and secure access to DEBs for new DEB-enabled applications as well as legacy applications.

The next section surveys the main design choices for meeting these goals. The subsequent sections describe an API for DEB-enabled applications, discuss how native applications can transparently access DEBs, and evaluate methods for securing DEB audit logs.

3.1 Design Choices

A format such as XML is a sensible choice for DEB files when they are stored outside a DEB-enabled file system. The original DEB specification [8] used plain text for the components of a DEB. However, a more efficient format is required for the native storage of DEBs. This is because unlike other compound file types, e.g., ZIP files or tarballs, DEBs are updated quite frequently as evidence units are introduced and the audit logs are modified. It is also likely that some DEBs will be extremely large, so methods for in-place updates will be necessary for efficient access. In summary, a native storage format for a DEB-enabled file system should support efficient updates of the audit log as well as rapid access to the digital evidence blobs.

Several file systems were evaluated before choosing a candidate for native DEB support, including ext2/3, reiserFS and XFS. NTFS was eliminated because its source code is not available, although NTFS alternate data streams are an attractive mechanism for implementing DEB resource forks (e.g., blobs of digital evidence, the audit log and DEB metadata). Most of the evaluated file systems contain features, e.g., extended attributes (EAs), that can be used to efficiently support DEBs. Unfortunately, the EA implementation in ext2/3 places substantial limits on the size of EAs (one disk block), precluding their use to store larger components in a DEB. Similarly, XFS and stable versions of reiserFS limit the size of EAs. In a future version of reiserFS, EAs will be stored as regular files in the file system, with the file name referring to

the name of the EA and the file contents being the associated value of the extended attribute. We adopt a similar strategy using symbolic links to store DEB components.

The best choice may be to use a standard file system like ext3. The resource forks within a DEB can be stored as separate files using symbolic links stored in the first data block of a DEB file. Changes are also required at the inode level (to tag DEBs as a special type of file and to accommodate efficient storage of DEBs), to pathname handling (to allow transparent access to digital evidence blobs within a DEB using a convention like `DEBname.blobname`), and at the system call level (to support a DEB API and legacy applications). At the system call level, standard file I/O calls such as `read()` and `write()` must be modified to perform auditing functions in addition to accessing blocks of a blob stored within a DEB.

We are currently using a user-level file system, FUSE (File System in User Space) [7], to test our ideas. System calls in FUSE are redirected by a kernel-level FUSE component to a user-space application (written against the FUSE library). This has enabled us to rapidly build a proof-of-concept primarily in user-space, without the complexity of in-kernel hacking.

3.2 API for DEB-Enabled Applications

This section describes an API for DEB-enabled applications to create, access and modify DEBs. The API's functions fall into three categories. The first category of functions facilitate the creation of DEBs and the introduction of "blobs" of digital evidence into DEBs. A blob is an arbitrary unit of digital evidence and might be a disk image, a single document or a compound file type. The second category allows access to a DEB's tags. Recall that the tags record DEB metadata, e.g., the investigating agent's name and contact information. The third category provides access to the DEB's audit log, so that applications can insert additional entries into the log to document investigative operations. Using any of the functions automatically introduces entries into the DEB's audit log. The functions are described briefly below.

- `int CreateDEB(char *filename, char *applicationinfo, char *comment, /* variable number of DEB tags */);`

This function creates a new DEB whose complete pathname is `filename`. The `comment` field is a free-form string entered into the audit log to describe the creation event, while `applicationinfo` documents the application that created the DEB. A variable number of tags, which document the investigator's name, contact infor-

mation, and case characteristics are permitted. An initial entry is made in the DEB's audit log to document the creation event. This entry also contains a hash of the initial DEB contents, which at this stage are essentially metadata. The `AddDEBBlob()` function, described below, allows evidence to be introduced into the bag. A positive return value indicates successful creation of the bag.

- `int AddDEBBlob(char *filename, char *blobname, void *blob, char *applicationinfo, char *comment);`

This function introduces a new piece of digital evidence, `blob`, named `blobname`, into the bag whose pathname is `filename`. The `blobname` must uniquely identify the piece of digital evidence in the DEB, otherwise an error is generated. The `comment` is introduced into the DEB's audit log to describe the digital evidence introduced, while `applicationinfo` documents the application itself. In addition, audit log entries are automatically written to document the cryptographic hash of the introduced evidence plus a hash of the entire bag contents after the introduction of the `blob` is completed. A positive return value indicates successful introduction of the `blob`.

- `int AddDEBBlobFile(char *filename, char *blobname, char *blobfilename, char *applicationinfo, char *comment);`

This function performs the same operations as `AddDEBBlob()`, except that the digital evidence that is introduced is contained in the file `blobfilename`, instead of in a block of memory.

- `int OpenDEBBlob(char *filename, char *blobname, int mode, char *applicationinfo, char *comment);`

This function returns a file handle attached to the blob `blobname` contained in the DEB identified by `filename`. The file handle is opened with read/write permissions described by `mode`, which has the same semantics as the mode parameter for the standard C `open()` function. The `applicationinfo` argument describes the application issuing the open command while the `comment` describes the open operation (from the opening application's perspective) in the DEB's audit log. A positive return value indicates success.

- **void CloseDEBBlob(int handle, char *comment);**

This function releases the file handle, **handle**, attached to a single blob of evidence in a DEB. The **comment** describes the close operation (from the closing application's perspective) in the DEB's audit log.

- **unsigned long long ReadDEBBlobBlock(int handle, void *data, unsigned long long len, char *comment);**

This function reads a block of data from the stream identified by **handle**. The **handle** must have been obtained from a call to **OpenDEBBlob()**. The length of the block to be read is **len**. The **comment** argument describes the read operation from the application's perspective. The function returns the number of bytes read.

- **unsigned long long WriteDEBBlobBlock(int handle, void *data, unsigned long long len, char *comment);**

This function writes a block of data to the stream identified by **handle**. This **handle** must have been obtained from a call to **OpenDEBBlob()**. The length of the block to be written is **len**. The **comment** argument describes the write operation from the application's perspective. The function returns the number of bytes written.

- **char *GetDEBTagValue(char *filename, char *tagname, char *applicationinfo, char *comment);**

This function returns a pointer to a string containing the value of the tag **tagname** associated with the DEB **filename**. The **applicationinfo** argument describes the application issuing the operation while **comment** describes the operation in further detail in the DEB's audit log. The function returns NULL if the tag's value cannot be returned.

- **int PutDEBTagValue(char *filename, char *tagname, char *applicationinfo, char *comment);**

This function creates (or modifies) the tag **tagname**, setting (or replacing) its value by **tagvalue** for the DEB **filename**. The **applicationinfo** argument describes the application issuing the operation while **comment** describes the operation in further detail in the DEB's audit log. A positive return value indicates successful modification of the tag.

- `int OpenDEBAuditLog(char *filename, char *application info, char *comment);`

This function returns a file handle associated with the audit log for the DEB `filename`. The file handle's mode is read-only. This primary use of the function is to review the audit log. To modify the audit log, the function `AppendDEBAuditLog()` must be invoked.

- `void CloseDEBAuditLog(int handle, char *application info, char *comment);`

This function closes the audit log stream associated with `handle`.

- `int AppendDEBAuditLog(char *filename, char *auditentry, char *applicationinfo, char *comment);`

This function appends a log entry `auditentry` to the audit log associated with the DEB `filename`. A positive return value indicates a successful append operation.

3.3 Support for Non-DEB-Enabled Applications

Native file system support for DEBs enables them to be used even with non-enabled applications. Rather than using the API described in Section 3.2, legacy applications may use standard C library `open()`, `close()`, `read()` and `write()` operations (and their buffered counterparts) on digital evidence blobs in a DEB. The `open()` system call is modified to return a handle to a blob in a DEB; operations against the returned handle target the associated digital evidence blob rather than the DEB itself. Hooks in the implementation of these system calls can identify the process name, process number and affected blocks, facilitating transparent updates of the audit log in the DEB. This information is useful not only in identifying which legacy applications have accessed the DEB, but also in auditing the behavior of a legacy application. For example, unauthorized write operations can be readily identified from the audit log. Also, the “thoroughness” of an application can be identified, by ensuring that it truly accesses all the blocks in a blob of digital evidence.

We have developed a prototype system for native file system support of non-DEB enabled applications based on FUSE. In our prototype, user-level applications are used to import and export DEBs into and out of a special DEB-aware FUSE file system. An import operation essentially splits the DEB into component files and places these files in a special directory, along with the DEB audit log and other metadata. Legacy access to digital evidence blobs in these special directories automatically

results in audit log updates. For example, read access to a digital evidence blob causes the application name (and process number), access time, portions of the blob accessed, and optionally, a hash of the executable of the accessing application to be recorded. The creation of a new blob of digital evidence results in a similar audit log entry. Exporting a DEB from the DEB-enabled file system simply recreates the DEB structure from the data stored in the corresponding directory.

Table 1. Scalpel JPG file carving results (1 GB disk image).

Scalpel v1.52	Time
File carving on ext3 file system (no legacy DEB support)	3 min. 12 sec.
File carving on ext3 file system (legacy DEB-enabled FS)	3 min. 29 sec.

We ran several experiments to determine the overhead of automatically auditing access to digital evidence blobs. Table 1 presents the results obtained when Scalpel was used to carve JPEG files from an 1 GB disk image. The test was run under Linux on a 1.6 GHz Pentium M Thinkpad with 2 GB of RAM.

Table 2. FTK evidence processing results (8 GB disk image).

FTK v1.60	Time
Add Evidence step on Samba share (no legacy DEB support)	47 min. 56 sec.
Add Evidence step on Samba share (legacy DEB-enabled FS)	59 min. 04 sec.

Table 2 shows the results obtained for FTK v1.60's Add Evidence step on an 8 GB disk image. FTK was run on a 3 GHz Pentium 4 desktop with 2 GB of RAM. Access to the DEB-enabled file system was through Samba over a 100 Mb Ethernet connection. The Samba server was a 1.7 GHz Thinkpad with 512 MB of RAM.

Under Linux, with direct access to the DEB file system, the overhead is approximately 9%. Over a Samba mount, FTK showed about 23% overhead, but further investigation indicated that the Windows XP platform running FTK was issuing two parallel, non-overlapping sequences of read operations through Samba, even when application accesses were strictly sequential. Running the Scalpel file carver under Windows over Samba to access a DEB-enabled file system showed similar overhead (approximately 25%; this result is not shown in the tables). We plan to investigate this strange Samba behavior in the future, but note that other developers have seen similar behavior in Windows XP.

Naturally, there are limitations to providing automatic auditing of DEB-unaware applications. For one, the audit log is not as “tidy” as it might be if auditing were controlled by a compliant application using the DEB API. This is because audit log entries for reads (or writes) that serve a common purpose cannot be easily grouped; since our prototype does not have high-level application knowledge, it can only track low-level file operations. Another limitation is that access to the special DEB directories via a network share, e.g., Samba, obfuscates the name of the application touching a blob of digital evidence. For example, if a Windows application accesses DEB data through a Samba share, the audit log shows the Samba daemon under Linux (*smbd*) as the accessing application. Still, we believe our legacy application support is a good interim solution as legacy applications are modified to use common DEB formats or are replaced with DEB-compliant applications.

3.4 Secure Audit Logs

To further strengthen DEB auditing capabilities, anti-tampering facilities can be introduced for DEB contents, especially the audit log. Our goal is not to prevent tampering of the audit log and DEB contents, but rather, to solve the slightly easier problem of detecting tampering. In general, secure auditing facilities require a trusted component. This component can be a WORM drive to which audit log entries are appended, or a secure server that is physically inaccessible to an attacker. In the following, we discuss some design choices.

Schneier and Kelsey [3] presented a scheme for secure auditing, which involves an untrusted machine U (e.g., a machine used in a digital investigation) that shares a secret A_0 with a trusted machine T . To append a new log entry D_j , U computes $K_j = \text{hash}(A_j)$, $C = E_k(D_j)$, $Y_j = \text{hash}(Y_{j-1}; C)$, and $Z_j = \text{MAC}_{A_j}(Y_j)$. Y_j is the j th entry in a hash chain, where $Y_1 = 0$ and MAC is a keyed hash function. Then, $[C, Y_j, Z_j]$ is written to the log. The shared secret is then recomputed: $A_{j+1} = \text{hash}(A_j)$, and A_j is destroyed. This scheme is tailored to disallow log entries created before a compromise at time t from being read by an attacker. The idea is that the attacker is then left to delete the entire log (which will be noticed when communication is established in the future between U and T) or leave the log alone (and not know if a log entry has recorded his unauthorized access). The scheme is useful if access to previous log entries by applications running on U is not required. Note that T can verify that the audit log on U is correct because it possesses A_0 and can “replay” the entire log.

Snodgrass *et al.* [6] have proposed a technique that allows read access to an audit log while preventing widespread tampering of the log. The scheme uses a trusted notary service, which accepts a digital document, computes a hash of the document and a secure timestamp and then stores and returns a notary ID. This notary ID is stored with the log entry. To determine if the audit log is consistent, a trusted party can verify that the notary IDs (and associated timestamps) on the notary service match those in the audit log. Omissions, additions and deletions can all be identified. This basic scheme has the drawback of requiring significant communication with the notary service, but audit log entries can be combined and submitted as a single document to the notary to reduce overhead (at the expense of a coarser level of log validation). The Snodgrass approach is particularly attractive for DEB audit logs as only limited storage is required on the trusted server. For each audit log entry, a hash is computed for the text of the log entry, this hash is submitted to the notary service, and the notary ID received is then stored in the DEB's audit log. Note that the DEB's audit log is readable by any application, which is useful for creating reports, evaluating an investigation, or performing tool evaluation.

4. Conclusions

Digital Evidence Bags (DEBs) mimic traditional evidence bags by providing a standard container for arbitrary digital evidence, with an integrated audit log and metadata that describes the evidence and the forensic processes applied to the evidence. Digital-forensics-aware operating system components – as provided by native file system support for DEBs – can significantly improve the performance and consistency of forensic investigations. The power of DEBs is increased substantially by providing a standard API and native file system support, because new applications (specifically written to support DEBs) and native applications (which use standard Unix system calls for I/O) can take advantage of automatic auditing of forensic operations.

Our system is a work in progress. However, once the initial implementation is stable, we expect to undertake a thorough performance study and determine whether user-level file system enhancements offer sufficient performance, or whether modifications to an existing file system, such as ext3, are actually necessary.

References

- [1] AccessData Corporation, Forensic Toolkit (FTK) (www.accessdata.com).

- [2] V. Roussev and G. Richard III, Breaking the performance wall: The case for distributed digital forensics, *Proceedings of the Fourth Digital Forensics Research Workshop*, 2004.
- [3] B. Schneier and J. Kelsey, Secure audit logs to support computer forensics, *ACM Transactions on Information and System Security*, vol. 2(2), pp. 159-176, 1999.
- [4] Sleuthkit.org, Autopsy (www.sleuthkit.org).
- [5] Sleuthkit.org, Sleuth Kit (www.sleuthkit.org).
- [6] R. Snodgrass, S. Yao and C. Collberg, Tamper detection in audit logs, *Proceedings of the Thirtieth International Conference on Very Large Databases*, pp. 504-515, 2004.
- [7] SourceForge.net, FUSE: Filesystem in user space (fuse.sourceforge.net).
- [8] P. Turner, Unification of digital evidence from disparate sources (digital evidence bags), *Proceedings of the Fifth Annual Digital Forensics Research Workshop*, 2005.

Chapter 4

REMOTE UPLOAD OF EVIDENCE OVER MOBILE AD HOC NETWORKS

Indrajit Ray

Abstract In this work, we report on one aspect of an autonomous robot-based digital evidence acquisition system that we are developing. When forensic investigators operate within a hostile environment they may use remotely operated unmanned devices to gather digital evidence. These systems periodically upload the evidence to a remote central server using a mobile ad hoc network. In such cases, large pieces of information need to be fragmented and transmitted in an appropriate manner. To support proper forensic analysis, certain properties must be ensured for each fragment of evidence – confidentiality during communication, authenticity and integrity of the data, and, most importantly, strong evidence of membership for fragments. This paper describes a framework to provide these properties for the robot-based evidence acquisition system under development.

Keywords: Evidence collection, authenticity, mobile ad hoc networks

1. Introduction

Consider the following scenario. A team of intelligence officers is trying to penetrate a terrorist network. The officers are working in a loosely coupled manner, geographically dispersed within hostile territory and far away from their command and control center. To limit their personal exposure to hostile activities and to enable evidence gathering from remote locations the team is using a number of unmanned vehicles (both aerial and terrestrial). These vehicles are network capable and equipped with various sensors (e.g., high resolution cameras, listening devices, chemical and bio-sensors) that gather information and periodically upload them to the central server at the command and control location. Later, forensic analysis is carried out on this data at the central location. Since the instruments need to be deployed in a fairly ad hoc manner at locations

Please use the following format when citing this chapter:

Ray, I., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 41–54.

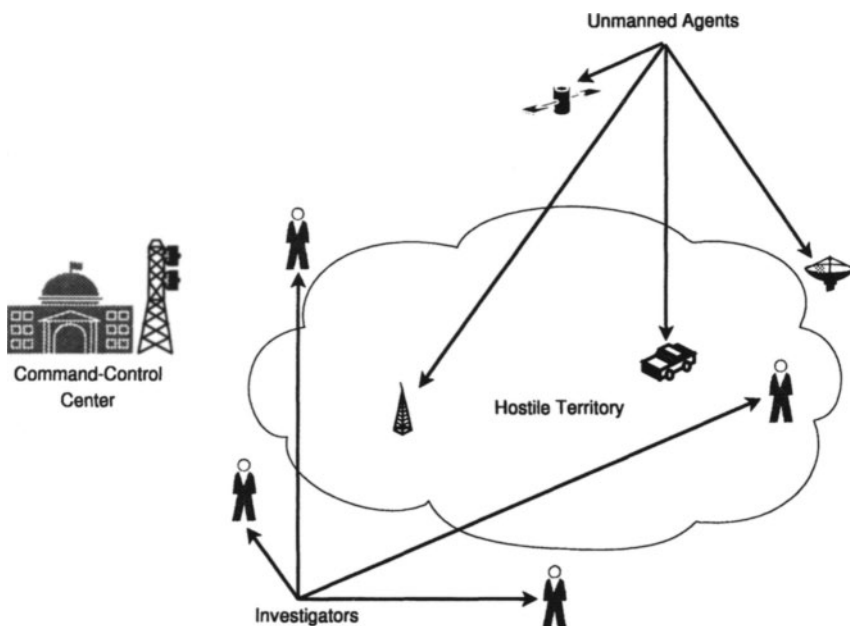


Figure 1. Target investigation environment.

that do not have access to a secure network infrastructure, they need to communicate securely with each other and with the central server over one or more mobile ad hoc networks (MANETs). A MANET can be deployed either solely over these devices or may use other mobile agents in the neighborhood. The instruments as well as the MANET(s) may be subject to active and passive attacks by adversaries. The environment is illustrated in Figure 1.

At Colorado State University, we are developing such an evidence acquisition system using mobile robots. Each mobile robot runs on a rechargeable battery and is equipped with a laptop computer that has wireless connectivity. The laptop computer runs the software that controls various sensors and actuators on the robot and provides support for ad hoc networking. Each robot is also equipped with a video camera (sensor) and a gripper arm (sensor/actuator) that can be remotely operated via the laptop. All the protocols described in this paper have been implemented and tested on a platform comprising three such robots.

Ensuring that the data uploaded by the unmanned devices is suitable for forensic analysis is not trivial. There are significant challenges imposed by MANETs that need to be addressed. First and foremost, authenticity and integrity of the data must be ensured during transmission. Since active attacks are possible, cryptographically strong schemes

must be provided for this purpose. Second, secure communication channels must be established between the various unmanned devices and the central server. The problem is that of ensuring secure group communication. The group, however, dynamically mutates, with the possibility that any of the communicating devices may be captured by the enemy and reverse engineered and/or destroyed. The group needs to be re-keyed periodically, ensuring that any protocol that is used for this purpose is efficient and is not compromised even when one or more of the devices are compromised. The third important challenge arises from the fact that MANETs are, in general, extremely unreliable, low bandwidth networks that provide no guarantees on message delivery. This means that to transmit large amounts of data, the data must be fragmented and there are no assurances that all fragments will be delivered. For the data to be useful as digital evidence, it is necessary to ensure that the fragments that have been received can be reconstructed to produce the original data, and to prove that each received fragment is part of the original data that was transmitted.

In this paper, we address these three issues in the following manner. We develop a protocol that allows us to deploy secure group communication over a MANET. This protocol is based partly on a theory of compatible keys that we have developed previously [9]. Next, we propose a scheme based on Rabin's Information Dispersal Algorithm (IDA) [8] that, together with conventional cryptographic checksums, allow us to reconstruct the original data from the fragments that have been received while ensuring authenticity and integrity. Finally, we describe a scheme based on one-way accumulator functions that allow us to prove the membership of fragments that are received over the MANET. Although some aspects of this problem have been solved individually, to our knowledge, this is the first scheme that provides a complete solution to the problem. We do not address the problem of routing and forwarding for MANETs as we assume that such a protocol already exists.

The rest of the paper is organized as follows. Section 2 addresses the issue of secure group communication over MANETs. Section 3 deals with the problem of preparing the evidence for reliable and authenticated upload over MANETs. In particular, Section 3.1 describes how a digest is prepared for the data and how the digest together with the data are fragmented using IDA. It also explains how a unique witness packet can be appended to each fragment to permit proofs of membership. Section 3.2 discusses the generation of fragment witnesses. Section 4 provides the conclusions.

2. Establishing Secure Group Communication

One of the biggest challenges for secure group communication is efficient group re-keying for dynamically mutating groups. Our group key protocol is designed to make re-keying simple and efficient. We begin by establishing the theory behind our group key protocol.

DEFINITION 1 The set of *messages* $\mathcal{M}$ is the set of non negative integers m that are less than an upper bound N , i.e.

$$\mathcal{M} = \{m \mid 0 \leq m < N\}. \quad (1)$$

DEFINITION 2 Given an integer a and a positive integer N , the following relationship holds,

$$a = qN + r \text{ where } 0 \leq r < N \text{ and } q = \lfloor a/N \rfloor \quad (2)$$

where $\lfloor x \rfloor$ denotes the largest integer less than equal to x . The value q is referred to as the *quotient* and r is referred to as the *remainder*. The remainder r , denoted $a \bmod N$, is also referred to as the *least positive residue* of $a \bmod N$.

DEFINITION 3 For positive integers a , b and N , a is *equivalent* to b , modulo N , denoted by $a \equiv b \bmod n$, if $a \bmod n = b \bmod n$.

DEFINITION 4 For positive integers a , x , n and $n > 1$, if $\gcd(a, n) = 1$ and $ax \equiv 1 \bmod n$, then x is referred to as the *multiplicative inverse* of a modulo n . Two integers a , b are said to be *relatively prime* if their only common divisor is 1, that is, $\gcd(a, b) = 1$. The integers $n_1, n_2, \dots, n_k$ are said to be *pairwise relatively prime*, if $\gcd(n_i, n_j) = 1$ for $i \neq j$.

DEFINITION 5 Euler's totient function $\phi(N)$ is defined as the number of integers that are less than N and relatively prime to N . Below we provide some relevant properties of totient functions.

1. $\phi(N) = N - 1$ if N is prime.
2. $\phi(N) = \phi(N_1)\phi(N_2) \dots \phi(N_k)$ if $N = N_1N_2 \dots N_k$ and $N_1, \dots, N_k$ are pairwise relatively prime.

THEOREM 1 Euler's theorem states that for every a and N that are relatively prime, $a^{\phi(N)} \equiv 1 \bmod N$, where $\phi(N)$ is Euler's totient function.

Proof: We omit the proof of Euler's theorem and refer interested readers to a book on number theory (see, e.g., [6]).

COROLLARY 1 If $0 < m < N$ and $N = N_1 N_2 \dots N_k$ and $N_1, N_2, \dots, N_k$ are primes, then $m^{x\phi(N)+1} \equiv m \pmod{N}$ where x is an integer.

DEFINITION 6 A *key* K is defined to be the ordered pair $\langle e, N \rangle$, where N is a product of distinct primes, $N \geq M$ and e is relatively prime to $\phi(N)$; e is the *exponent* and N is the *base* of the key K .

DEFINITION 7 The *encryption* of a message m with the key $K = \langle e, N \rangle$, denoted as $[m, K]$, is defined as

$$[m, \langle e, N \rangle] = m^e \pmod{N}. \quad (3)$$

DEFINITION 8 The *inverse* of a key $K = \langle e, N \rangle$, denoted by K^{-1} , is an ordered pair $\langle d, N \rangle$, satisfying $ed \equiv 1 \pmod{\phi(N)}$, where $\phi(N)$ is Euler's totient function.

THEOREM 2 For any message m .

$$[[m, K], K^{-1}] = [[m, K^{-1}], K] = m \quad (4)$$

where $K = \langle e, N \rangle$ and $K^{-1} = \langle d, N \rangle$.

Proof: We first show that

$$\begin{aligned} & [[m, K], K^{-1}] = m \\ L.H.S. &= [[m, K], K^{-1}] \\ &= [m^e \pmod{N}, K^{-1}] \\ &= (m^e \pmod{N})^d \pmod{N} \\ &= m^{ed} \pmod{N} \\ &= m^{(x\phi(N)+1)} \pmod{N} \\ &= m \pmod{N} \\ &= m \\ &= R.H.S. \end{aligned}$$

By symmetry $[[m, K^{-1}], K] = m$.

COROLLARY 2 An encryption, $[m, K]$, is *one-to-one* if it satisfies the relation

$$[[m, K], K^{-1}] = [[m, K^{-1}], K] = m.$$

DEFINITION 9 Two keys $K_1 = \langle e_1, N_1 \rangle$ and $K_2 = \langle e_2, N_2 \rangle$ are said to be *compatible* if $e_1 = e_2$ and N_1 and N_2 are relatively prime.

DEFINITION 10 If two keys $K_1 = \langle e, N_1 \rangle$ and $K_2 = \langle e, N_2 \rangle$ are compatible, then the *product* key, $K_1 \times K_2$, is defined as $\langle e, N_1 N_2 \rangle$.

LEMMA 1 For positive integers a , N_1 and N_2 ,

$$(a \bmod N_1 N_2) \equiv a \bmod N_1.$$

Proof: Let $a = N_1 N_2 x + N_1 y + z$, where x , y and z are integers.

$$\begin{aligned} L.H.S. &= (a \bmod N_1 N_2) \bmod N_1 \\ &= \left(N_1 N_2 x + N_1 y + z - \left\lfloor \frac{N_1 N_2 x + N_1 y + z}{N_1 N_2} \right\rfloor N_1 N_2 \right) \bmod N_1 \\ &= (N_1 y + z) \bmod N_1 \\ &= z \end{aligned}$$

$$\begin{aligned} R.H.S. &= (a \bmod N_1) \\ &= (N_1 N_2 x + N_1 y + z) \bmod N_1 \\ &= z \end{aligned}$$

Hence the proof.

THEOREM 3 For any two messages m and $\hat{m}$, such that $m, \hat{m} < N_1, N_2$,

$$[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \bmod N_1 \text{ if and only if } m = \hat{m} \quad (5)$$

$$[m, K_1 \times K_2] \equiv [\hat{m}, K_2] \bmod N_2 \text{ if and only if } m = \hat{m} \quad (6)$$

where K_1 is the key $\langle e, N_1 \rangle$, K_2 is the key $\langle e, N_2 \rangle$ and $K_1 \times K_2$ is the product key $\langle e, N_1 N_2 \rangle$.

Proof: The proof for (6) is similar to that for (5). Therefore, we only provide the proof for (5).

[If part] Given $m = \hat{m}$, prove that $[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \bmod N_1$, that is,

$$\begin{aligned} [m, K_1 \times K_2] \bmod N_1 &= [\hat{m}, K_1] \bmod N_1 \\ L.H.S. &= [m, K_1 \times K_2] \bmod N_1 \\ &= (m^e \bmod N_1 N_2) \bmod N_1 \\ &= m^e \bmod N_1 \\ R.H.S. &= [\hat{m}^e \bmod N_1] \bmod N_1 \\ &= \hat{m}^e \bmod N_1 \\ &= m^e \bmod N_1 \end{aligned}$$

[Only If part] Given $[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \pmod{N_1}$, we have to prove $m = \hat{m}$

$$\begin{aligned}
 [m, K_1 \times K_2] &\equiv [\hat{m}, K_1] \pmod{N_1} \\
 \text{or } [m, K_1 \times K_2] \pmod{N_1} &= [\hat{m}, K_1] \pmod{N_1} \\
 \text{or } (m^e \pmod{N_1 N_2}) &= (\hat{m}^e \pmod{N_1}) \\
 \text{or } m^e \pmod{N_1} &= (\hat{m}^e \pmod{N_1}) \pmod{N_1} \\
 \text{or } [m, \langle e, N_1 \rangle] &= [\hat{m}, \langle e, N_1 \rangle] \\
 \text{or } m &= \hat{m}
 \end{aligned}$$

The more general case for the above theorem is as follows.

$$[m, K_1 \times K_2 \dots K_p] \equiv [\hat{m}, K_1] \pmod{N_1} \text{ if and only if } m = \hat{m} \quad (7)$$

$$[m, K_1 \times K_2 \dots K_p] \equiv [\hat{m}, K_1] \pmod{N_2} \text{ if and only if } m = \hat{m} \quad (8)$$

⋮

$$[m, K_1 \times K_2 \dots K_p] \equiv [\hat{m}, K_1] \pmod{N_p} \text{ if and only if } m = \hat{m} \quad (9)$$

where $K_1 = \langle e, N_1 \rangle$, $K_2 = \langle e, N_2 \rangle$, etc. and $K_1 \times K_2 \dots K_p$ is the product key $\langle e, N_1 N_2 \dots, N_p \rangle$.

DEFINITION 11 The *group key* K_g for a group of k members $p_1, \dots, p_k$, with a member p_i having the public/private key pair $K_i = \langle e, N_i \rangle / K_i^{-1} = \langle d_i, N_i \rangle$, is defined to be the product key $K_1 \times K_2 \times \dots \times K_k$.

COROLLARY 3 A message $m < N_1 \times N_2 \dots N_k$ encrypted with a group key $K_g = K_1 \times K_2 \times K_k$ can be decrypted with any one of the private keys K_i^{-1} .

Proof: The proof follows directly from Lemma 1 and Theorem 3.

2.1 Group Key Generation and Revocation

Group key management is usually done at the central server. This involves initial group key generation and periodic re-keying of the group. Re-keying to exclude a group member can also be performed at the local level. This can be done when one or more agents identify that one of their neighbors have been compromised. We employ re-keying by a local agent as one of the many ways to signal the command and control center that an agent has been compromised. In the following sections we discuss key generation and revocation in more detail.

Initial Group Key Generation: The group key is initially generated at the command and control center.

1. The center chooses a server key pair $\langle K_s, K_s^{-1} \rangle$ for itself and key pairs $\langle K_i, K_i^{-1} \rangle$ for the unmanned devices that will be deployed. These keys are compatible to each other according to Definition 9.
2. The group key is generated as $K_{gr} = K_s \times K_1 \times \dots \times K_n$. Each unmanned device stores the key pair $\langle K_{gr}, K_i^{-1} \rangle$. The keys $K_s, K_s^{-1}, K_1 \dots K_n, K_{gr}$ are stored at the central server.
3. All entities use the group key K_{gr} to encrypt messages. To decrypt a message, an unmanned device uses the key K_i^{-1} while the server uses K_s^{-1} . An agent uses K_i^{-1} to sign a message, while the server uses K_s^{-1} .

Group Re-Keying: The group key must be changed every time the group changes and it should also be changed periodically to limit exposure of the key. The group changes most often when an agent is compromised. An agent compromise is suspected by the central server if it does not receive an “alive” message from the agent for a period of time. At this stage, it takes one of two steps. It either determines unilaterally that the agent is compromised (it may have other information that confirms the suspicion) or it instructs other agents in the suspected agent’s neighborhood to confirm the suspicion. The neighboring agents then execute a consensus algorithm among themselves to validate the central server’s suspicion. Re-keying for the different scenarios takes place as follows.

1. To exclude an agent, let the agent which needs to be excluded from the group be agent i . The group key $K_{gr} = K_s \times K_1 \times \dots \times K_i \times \dots \times K_n$ is divided by K_i to generate the new group key as $K'_{gr} = K_s \times K_1 \times \dots \times K_{i-1} \times K_{i+1} \times \dots \times K_n$. The new group key is digitally signed by the server and sent to the remaining agents encrypted under itself. Thus, all the agents except the compromised one are able to obtain the new encryption key. (This will be true for every agent j as long as K_j is a factor of the group key.)
2. To include a new agent $n + 1$ within the group, the central server creates the new group key $K'_{gr} = K_s \times \dots \times K_n \times K_{n+1}$. This key is digitally signed by the server, encrypted under the new group key itself and forwarded to the remaining agents.

3. To perform periodic re-keying, the new group is generated according to the group key generation process. For each agent i a new key pair $\langle Q_i, Q_i^{-1} \rangle$ is generated. The relevant information is sent encrypted under the agent's previous key K_i .

2.2 Security and Implementation Issues

The group key system is as secure as the RSA cryptosystem. The most damaging attack on this scheme would be for an attacker to discover the private key K_i^{-1} corresponding to the key K_i involved in the group key K_g . This will enable the attacker to decrypt all messages encrypted with the group key. The obvious way to do this is to factor the modulus N_i of K_i into its two prime factors. This is a well known hard problem.

Another way to break the group key encryption is to find a technique to compute the e^{th} roots mod N . Since the encrypted message is $c = m^e \bmod N_1 \times N_2 \times \dots \times N_k$ and $N = N_1 \times N_2 \times \dots \times N_k$ is publicly known, the e^{th} root of $c \bmod N_1 \times N_2 \times \dots \times N_k$ is the message m . This attack is not known to be equivalent to factoring. However, as pointed out in [10], there are no known techniques for performing this attack.

Care must be taken when choosing the exponent e . If e is small and $m^e < N_1 \times N_2 \times \dots \times N_k$, then $c = m^e \bmod N_1 \times N_2 \times \dots \times N_k = m^e$. Then, by simply extracting the e^{th} root of c an attacker can obtain the message m . Choosing a large e makes the encryption process computationally intensive. On the other hand, choosing an appropriate algorithm for the computation, alleviates the problem considerably. For example, the best time bound for multiplying two n -bit integers is $O(n \log n \log \log n)$ [3]. Finding $m^e \bmod p$, where p is an n -bit integer, is $O(\log e \ n \log n \log \log n)$.

3. Evidence Upload

Since a mobile ad hoc network is highly unreliable, an agent intending to upload a piece of evidence to the main server has to transmit the message in a redundant manner. To do this, the agent can transmit the entire evidence multiple times. However, this is expensive in terms of bandwidth and power consumption. Rabin's Information Dispersal Algorithm (IDA) [8] can be used to fragment data and transmit it efficiently. The fragmentation is performed so that if the original evidence has n fragments, the receipt of any m fragments at the central server ensures that all the evidence is received. Note that this scheme is cost effective as well as bandwidth efficient.

Data fragmentation raises an important issue: a strong proof of membership is required for all the fragments. In particular, it is necessary to

prove that every fragment that is received correctly belongs to the original message. The proof should be performed independently for each fragment. Consequently, this precludes the use of cryptographic techniques, e.g., Merkle hashes [5], where the hash (or signature) of a fragment is dependent on other fragments. In the following, we propose a novel scheme based on one-way accumulator functions to generate “witnesses” for each fragment. A witness serves as a proof of membership for a fragment. The algorithm for generating witnesses is described below.

3.1 Evidence Fragmentation

Let M be the message that has to be sent to the central server over the ad hoc network. In the following, $h(X)$ denotes a digest of a message X created by, say, SHA-1 or MD5; $S_{kprv}(X)$ denotes a signature on message X using a private key $kprv$; and $E_{kpub}(X)$ denotes an encryption of message X with a public key $kpub$. The signature is verified using an encryption algorithm. The “witness” corresponding to a particular message i is denoted by w_i . We assume that the computations are done in $GF(2^8)$. An agent intending to upload a message proceeds as follows.

1. The agent creates a message digest of the original message and digitally signs it with its private key $kprv$. The signed message digest is appended to the original message to get the new message $M' = M \| S_{kprv}(h(M))$. Let the length of the new message be N .
2. The agent breaks M' into $\frac{N}{m}$ fixed sized fragments of length m such as $M' = (b_1 \dots b_m), (b_{m+1} \dots b_{2m}), \dots (b_{N-m+1} \dots b_N)$ where b_i are integers taken from a certain range $[0 \dots (2^r - 1)]$. That is, $M' = B_1 \| B_2 \| \dots \| B_{\frac{N}{m}}$ where $B_i = (b_{i-1m+1} \dots b_{im})$, $1 \leq i \leq \frac{N}{m}$. The size of each fragment needs to be smaller than the MTU of the ad hoc network.
3. The agent chooses a set A of n vectors $A = (\vec{A}_1, \vec{A}_2, \dots, \vec{A}_n)$ such that each $\vec{A}_i = (a_{i1}, a_{i2}, \dots, a_{im})$ and every subset of m different vectors in A are linearly independent.
4. M' is next processed and divided into n pieces $M'_1, M'_2, \dots, M'_n$ such that $M'_i = (\vec{A}_i \bullet B_1, \dots, \vec{A}_i \bullet B_{\frac{N}{m}})$, $i = 1 \dots n$, where $\vec{A}_i \bullet B_k = (a_{i1} \cdot b_{(k-1)m+1} + \dots + a_{im} \cdot b_{km})$.
5. For each M'_i the agent prepares a witness w_i as described in Section 3.2. Each witness w_i is appended to the corresponding M'_i . The agent also generates a single accumulated witness W from the

individual witness w_i . Then, following a process similar to one described in Steps 2–4, the agent disperses the accumulated witness W over the M'_i fragments. Let $\bar{W}_i$ denote the portion of W that is dispersed over M'_i . After this step each fragment F_i is given by: $F_i = M'_i \| w_i \| \bar{W}_i$.

6. A digest of each F_i is prepared, digitally signed by the agent's private key k_{prv} , and appended to fragment $F'_i = F_i \| S_{k_{prv}}(h(F_i))$.
7. Finally, each F'_i is encrypted with the group key established earlier and transmitted over the MANET to the destination.

We can easily show that if any m fragments F'_i are received correctly by the central server (i.e., F_i is received correctly) then

1. the original message M' can be reconstructed, and
2. the accumulated witness W can be reconstructed.

We omit the proof for lack of space. Interested readers are directed to Rabin's work [8] for details. The ratio n/m is dependent on the estimated loss rate of the MANET. Since each fragment is accompanied by a strong cryptographic checksum that is digitally signed, the authenticity of the fragment is ensured in transit. Additionally, since the message M' can be reconstructed, it follows that the signed digest $S_{k_{prv}}(h(M))$ is available to verify authenticity of message M .

3.2 Witness Generation

Let M'_r be the reconstructed message from the previous section. Each witness w_i together with the accumulated witness W are used to prove that each fragment M'_i is part of the same data from which the message M' has been reconstructed. We discuss how the different witnesses are created.

Witnesses are generated using one-way accumulator functions [2]. Briefly, a one-way accumulator is a special type of one-way hash function that satisfies the *quasi-commutative* property. A function $f : X \times Y \rightarrow X$ is said to be quasi-commutative if for all $x \in X$ and for all $y_1, y_2 \in Y$, $f(f(x, y_1), y_2) = f(f(x, y_2), y_1)$. An interesting consequence of this property is that if one starts with an initial value x and a set of values $y_1, y_2, \dots, y_n$, then the accumulated hash $z = f(f(f(\dots f(f(f(x, y_1), y_2), y_3), \dots y_{n-2}), y_{n-1}), y_n)$ would be unchanged if the order of the y_i 's were permuted.

Let us assume that such a one-way accumulator function f is available. Each M'_i that was obtained in the previous section after IDA was applied

to the message M' is processed as follows to generate the individual witness w_i and the accumulated witness W .

1. A unique seed value x_0 is defined globally and is available to every agent.
2. Using the one-way accumulator function f the agent calculates $x_i = f(x_{i-1}, M'_i)$ for the different M'_i . The final x_n is digitally signed by the agent and forms the accumulated witness W , i.e., $W = x_n || S_{kprv}(x_n)$.
3. For each M'_i the agent computes the witness w_i as

$$w_i = f(f(\dots f(f(x_0, M'_1), \dots, M'_{i-1}), M'_{i+1}) \dots, M'_n)$$

We can easily show that $x_n = f(w_i, M'_i)$. This proves that each received M'_i is a member of the original message from which M' was computed.

Benaloh and Mare were the first to propose the notion of one-way accumulator functions [2]. Subsequently, a number of researchers have proposed modifications to the original Benaloh-Mare proposal [1, 4, 7, 11, 12]. For the function f , we adopt an efficient implementation of the Benaloh-Mare scheme based on *universal-2* hash functions that has been proposed by Goodrich and co-workers [4].

4. Conclusions

The protocol presented in this paper permits digital evidence to be uploaded over mobile ad hoc networks in a secure and reliable manner. The data is transmitted in a manner that ensures its confidentiality, integrity and authenticity. The low bandwidth characteristics of a MANET necessitate data fragmentation. This, in turn, imposes the need for proofs of membership for the data fragments. A new group key protocol is devised to ensure secure communication over a MANET. Rabin's Information Dispersal Algorithm (IDA) is used to fragment data so that any m of n fragments allow the complete reconstruction of original data. One-way accumulator functions are used to generate proofs of membership for fragments. One component of the proof is dispersed over fragments to ensure that it will be always available when needed.

To our knowledge, this is the first integrated scheme that provides all the desired security and reliability properties. The protocol has been implemented as part of an on-going project focused on developing a robot-based digital evidence acquisition system. We hope to publish

other results from this project, including performance evaluation, in the near future.

References

- [1] N. Baric and B. Pfitzmann, Collision-free accumulators and fail-stop signatures without trees, *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (Lecture Notes in Computer Science, Volume 1233)*, Springer, Heidelberg, Germany, pp. 480-494, 1997.
- [2] J. Benaloh and M. de Mare, One-way accumulators: A decentralized alternative to digital signatures, *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (Lecture Notes in Computer Science, Volume 765)*, Springer, Heidelberg, Germany, pp. 274-285, 1993.
- [3] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, McGraw-Hill, Boston, Massachusetts, 2001.
- [4] R. Gennaro, S. Halevi and T. Rabin, Secure hash-and-sign signatures without random oracle, *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (Lecture Notes in Computer Science, Volume 1592)*, Springer, Heidelberg, Germany, pp. 123-139, 1999.
- [5] R. Merkle, A certified digital signature, *Proceedings of the Ninth Annual International Cryptology Conference (Lecture Notes in Computer Science, Volume 435)*, Springer, Heidelberg, Germany, pp. 234-246, 1989.
- [6] I. Niven and H. Zuckerman, *An Introduction to the Theory of Numbers*, John Wiley, New York, 1980.
- [7] R. Nyberg, Fast accumulated hashing, *Proceedings of the Third International Workshop on Fast Software Encryption (Lecture Notes in Computer Science, Volume 1039)*, Springer, Heidelberg, Germany, pp. 83-87, 1996.
- [8] M. Rabin, Efficient dispersal of information for security, load balancing and fault tolerance, *Journal of the ACM*, vol. 36(2), pp.335-348, 1989.
- [9] I. Ray, E. Kim, R. McConnell and D. Massey, Reliably, securely and efficiently distributing electronic content using multicasting, *Proceedings of the Sixth International Conference on E-Commerce and Web Technologies (Lecture Notes in Computer Science, Volume 3590)*, Springer, Heidelberg, Germany, pp. 327-336, 2005.

- [10] RSA Laboratories, RSA Laboratories' frequently asked questions about today's cryptography (v. 4.1) (www.rsasecurity.com/rsalabs), 2004.
- [11] T. Sander, Efficient accumulators without trapdoors, *Proceedings of the Second International Conference on Information and Communications Security (Lecture Notes in Computer Science, Volume 1726)*, Springer, Heidelberg, Germany, pp. 252-262, 1999.
- [12] T. Sander, A. T-Shma, and M. Yung, Blind auditable membership proofs, *Proceedings of the Fourth International Conference on Financial Cryptography (Lecture Notes in Computer Science, Volume 1962)*, Springer, Heidelberg, Germany, pp. 53-71, 2001.

Chapter 5

APPLYING MACHINE TRUST MODELS TO FORENSIC INVESTIGATIONS

M. Wojcik, H. Venter, J. Eloff and M. Olivier

Abstract Digital forensics involves the identification, preservation, analysis and presentation of electronic evidence for use in legal proceedings. In the presence of contradictory evidence, forensic investigators need a means to determine which evidence can be trusted. This is particularly true in a trust model environment where computerised agents may make trust-based decisions that influence interactions within the system. This paper focuses on the analysis of evidence in trust-based environments and the determination of the degree to which evidence can be trusted. The trust model proposed in this work may be implemented in a tool for conducting trust-based forensic investigations. The model takes into account the trust environment and parameters that influence interactions in a computer network being investigated. Also, it allows for crimes to be reenacted to create more substantial evidentiary proof.

Keywords: Trust models, forensic investigations, digital evidence

1. Introduction

Digital forensics involves the identification, preservation, analysis and presentation of electronic evidence for use in legal proceedings [1, 10, 14]. Clearly, digital evidence must be trustworthy for it to have any probative value in a courtroom. However, a dilemma arises when an investigator encounters evidence with varying interpretations, some of which contradict each other. In such an instance, a means is needed for determining which evidence can be trusted.

The problem is especially critical when the network containing the evidence in question is running some form of trust model architecture. Such a network allows computerised agents to participate in transactions on behalf of a user to find the most efficient way to conduct these interactions. Thus, it is possible that some of the files (especially sys-

Please use the following format when citing this chapter:

Wojcik, M., Venter, H., Eloff, J., Olivier, M., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 55–65.

tem files) that may look suspect were in actuality created by the agents executing the trust model and not by human users.

This paper proposes a trust-based model consisting of three phases to address the dilemma. A tool based on this trust model can be run by an investigator to determine the trustworthiness of network nodes and the influence of the trust environment on the files that are created. The term “nodes” in this context denotes devices running a trust model. The trust-based forensic model helps evaluate evidence to determine which evidence can be trusted and which evidence has been tampered with. It also allows for a crime to be reenacted and the evidence to be recreated to produce evidentiary proof that is complete, reliable and believable when admitted in court [14].

This paper is organised as follows. The next section, Section 2, provides an overview of trust models. Section 3 describes the proposed model that integrates key concepts from trust models and digital forensics to enhance digital forensic investigations. Section 4 presents the advantages and shortcomings of the model, and Section 5 provides concluding remarks.

2. Trust Models

New technologies have changed the business world to such an extent that even methods for establishing trust during business transactions have had to be revised to keep up with how transactions are performed. This has led to the formulation of trust models.

Trust is an abstract concept, the exact definition of which is unique for every individual. Trust relies on the formulation of templates for similar situational experiences. This allows an individual to group various experiences and their associated trust representations.

Nooteboom [11] defines trust as a four-place predicate: “Someone has trust in something, in some respect and under some conditions.” The individuals participating in a trust relationship in the context of trust models are called agents. Agents, in our work, refer to non-human, coded entities. These coded entities are defined by a programmer and embody logical rules [7] and restrictions against which interactions are analysed and processed to obtain a trust value. A trust value, which is calculated by a trust model, indicates the level of trust one agent has in another. The exact values that indicate trust, distrust and partial trust depend on the specific trust model. The “someone” and “something” in Nooteboom’s predicate refer to two agents participating in an interaction. Each agent has some form of trust in the other. The respect under which the trust is given refers to the situational factors that instigated

the need for the transaction and the conditions refer to the limitations under which the transaction occurs.

Trust models [3, 4, 9, 12] are used to analyse the trustworthiness of other agents. This includes the trustworthiness of information shared by agents, since this information is often used to make important decisions.

Trust values are obtained and assigned in various ways. Dynamic means of evaluating an agent and calculating a trust value include observation, experience and negotiation. Observation allows an agent to examine the interactions of other agents before attempting an interaction. Direct experience allows an agent to participate in an interaction and analyse the outcome [5]. Negotiation, on the other hand, requires that two agents share trust-related information contained in their security policies before commencing an interaction [8].

The result of the trust analysis process is a trust value that is used to restrict an interaction. In particular, it limits the information that is shared and it defines the behaviour of the interaction. Higher trust values result in freer interactions and higher trust in the information shared during the interactions.

Since the trust model influences how interactions are conducted, it also influences how information about the interactions is stored. This has a direct influence on forensic investigations because it influences potential evidence of criminal activity. Trust models are also able to determine which nodes are suspect in the trust environment. Such nodes are given distrust values based on their behaviour.

3. Defining Trust in Forensics

An investigator needs to know which evidence can be trusted in order to make sound judgments. This is an issue because criminals may attempt to tamper with evidence to affect its trustworthiness. Tampering, which includes planting false evidence, modifying data or deleting files, may impede evidence gathering as well as evidence analysis.

An investigator looks for anomalies, failures and specific results when running tests on a system. If the information has been tampered with to the extent that anomalies, failures and specific results are not discernible, an investigation can be led away from the source of criminal activity [13]. It is easiest to tamper with evidence contained in user-created files, which are easier to locate, understand and modify. System files often contain a wealth of information. However, system files are typically in obscure locations and protected by the operating system; tampering with these files requires specialised technical knowledge.

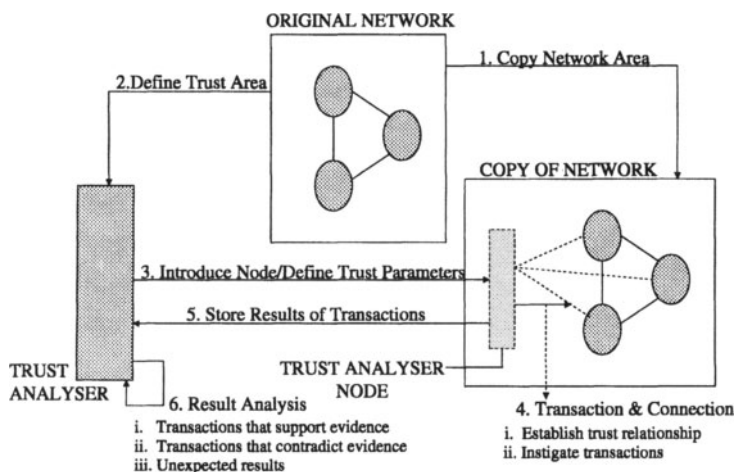


Figure 1. Using trust models to gather forensic evidence.

In a network that relies on a trust architecture, system files are created according to trust rules that govern the processing of interactions. Trust models define the level of trust given to agents participating in interactions. A system file that is created or modified during an interaction between two agents depends on the trust level assigned to the agents and on the nature of the interaction.

The results of these interactions update the state of trust in the system, influencing the processing of future interactions [6, 16]. Keeping this in mind, it is possible to test for criminal activities that have occurred over a network by testing the state of the trust relationships within the network and the reactions of various nodes to similar activities.

Figure 1 presents a scheme for using trust models to determine the presence and trustworthiness of forensic evidence. The numbered and labelled arrows in Figure 1 indicate processes that occur on four logical components. The first component is the original network in which devices containing evidence of criminal activity exist.

The second logical component is the copy of the original network. This is made by copying data from suspect devices and the network. To preserve the trust environment, the copy includes copies of devices that surround the suspect devices. It is important to note that only certain portions of the original network are copied for simulation and reenactment purposes. Therefore, it is important that these portions be carefully selected to include all the devices that may have influenced the

criminal activity. Each device will have files, usually some form of text files created by a particular trust model, that relate to the trust model in place. These files determine how each device reacts to trust-based interactions. The copy also contains the current state of these files.

A copy of the original state of system files (before the trust model was run) can assist in predicting the behaviour of the system and its new state. Although these files could contain trust-related data that may have been tampered with, the investigator should realise that any interaction that has occurred influences all the agents that participated in the interaction. For instance, tampering is to be suspected if one agent that participated in an interaction is found to trust another agent unconditionally while the other agents involved in the interaction show levels of distrust. To leave no traces, a criminal entity would have to tamper with all the agents that may have directly or indirectly participated in the interaction.

The third and fourth logical components, the trust analyser and a logical trust analyser node, make up one physical component, the trust analysis unit. The trust analyser is an investigation tool to be used in trust environments when the influence of the prevailing trust environment is to be determined. The logical trust analyser node is created by the trust analyser and is introduced into the copy of the network to act as an additional node in the network.

The proposed trust model for digital forensics has three phases: the establishment phase, the evidence gathering phase and the analysis phase. The first three processes in the Figure 1 are part of the establishment phase. Processes 4 and 5 are part of the evidence gathering phase, and Process 6 is a phase on its own (analysis phase). The establishment phase sets up the necessary criteria and environmental variables to conduct an investigation. The evidence gathering phase actively gathers evidence for analysis, while the analysis phase produces a conclusion based on the evidence.

3.1 Establishment Phase

The establishment phase begins with the identification of evidence. This phase is paramount as it influences the progress and results of all the following phases [14].

Once an investigator has identified what evidence is present and how it is stored, the evidence must be isolated and collected. This process must not damage or cause any loss of evidence. Also, it should allow for the evidence to be analysed to acquire the relevant information [15].

The first step required by the model is to copy the data and network settings from suspect devices. This is done to preserve the original state of the evidence. The model makes use of distributed computing to ease the computational load. This is implemented by duplicating the network on several machines to simulate the “live” state of the network on a so-called “dead” copy. However, it is not necessary to have a physical machine for every physical machine in the target network. Several sub-sections of the network can be duplicated on a single machine. Suspect nodes are placed on their own computer; nodes that support suspect nodes are grouped with the nodes they support to create a sub-domain. The use of multiple machines allows for the duplication of some of the more vital physical links. Also, it provides for a more accurate representation of the live state of the original network for subsequent analysis.

After the network area of interest has been duplicated, an analysis is conducted to determine the appropriate trust attributes. One way of determining the trust attributes is to query the people involved with network setup. Should this not be possible, the information can be gleaned from the network itself by searching for global policies that have been defined. The investigator should also be able to directly access the list of rules governing trust from any of the physical nodes in the network. Whether trust is built by reputation, observation or direct interaction with the new nodes depends on the rules that influence the prevailing trust environment. This process must be done with as much, if not more, care as making the copy of the system: it is important that no changes are made to the system state while extracting information. The rules are input into the trust analyser as text documents.

Various activities linked to the suspected crime must be defined as a set of attributes. A crime involving an information leak could have attributes corresponding to the manner in which the information was leaked and the confidentiality level of the leaked information. These could be represented as values and logical rules. For instance, the confidentiality level of information could be a set of values and the means by which the information was leaked can be indicated as parameters. For example, if email is the medium for leakage, the parameters would be the sender’s and recipient’s addresses. The trust analyser uses these attributes to attempt to recreate the crime.

The trust analyser uses the rules and attributes to define a virtual node for the network that runs according to the same rules and attributes defined by the network and trust environment. This virtual node is then introduced into the copy of the network where it is required to run and gather trust-related information.

3.2 Evidence Gathering Phase

The goal of the evidence gathering phase is to obtain information of value to a forensic investigation. The evidence must be gathered under the restrictions placed by the establishment phase. The virtual node introduced into the copy of the original network controls this process.

The driving force of the evidence gathering phase is the transaction and connection process, which is made up of two key sub-processes: trust establishment and transactions. Trust establishment takes into consideration the trust area and trust parameters received from the establishment phase. It uses this information to establish communication links with the other nodes. This establishes the trust levels between nodes and ensures that the new node is governed by the same context as the original nodes in the suspect network. The new node, therefore, instigates transactions in the same manner as nodes in the original network.

To successfully recreate the evidence, it is important to have a clear definition of the suspected crime and the context in which the crime occurred. Both of these factors are derived from the establishment phase. Once the trust context has been established, the virtual node conducts a detailed analysis of the attributes that are related to the suspected crime. These attributes are used to deduce interactions that should have occurred for the suspected criminal activity to take place.

The transactions sub-process makes use of the already-established trust connections to recreate the forensic evidence that is being questioned. It involves the recreation of events that created the suspect evidence to test whether the results correlate with the suspected crime. For example, the trust analyser may attempt to send confidential information outside the network and examine how this behaviour changes the trust environment.

The responses of the system to the various transactions are recorded and passed back to the trust analyser node. After all the transactions have been finalised, data created by the various nodes, including that created by the virtual node, is collected and returned to the trust analyser for detailed analysis, which occurs in the final (analysis) phase. This data is representative of the system's final state after the transactions have occurred and is used for comparisons with the original evidence.

3.3 Analysis Phase

During the analysis phase, the results are gathered and investigated to reach a conclusion. The trust analyser is supplied with machine-generated data created by the nodes in the network as a result of the

transactions instigated by the virtual trust analyser node. This machine-generated data is compared with other machine-created data that constitutes evidence of a specific crime.

The results are analysed along with the trust rules of the system to determine how the prevailing trust environment influences the representation of the collected evidence. The influence is taken into consideration during the more detailed analysis phase. The analysis may produce one or a combination of three different sets of results: results supporting the evidence, results contradicting the evidence and unexpected results. Various conclusions may be drawn from these results.

The results that support the evidence and contradict the evidence are dependent on the fact that the investigator is expecting certain evidence to correlate and other evidence to contradict. If the results are what the investigator expects, he/she has a means of proving that the suspicions are true. Results that correlate are indicative of a successful recreation of a crime and can be used with the original evidence to prove that the suspected crime did indeed occur. Results that are expected to be contrary, perhaps due to a suspicion that data was tampered with, also support a given theory.

Unexpected results can be scrutinised in two ways depending on an investigator's initial outlook. These results include those that were expected to correlate and do not, and those that were expected to be contradictory but in fact correlate. If the investigator believes his/her theory to be sound and is certain about what results would support the theory, unexpected results could mean that the investigator's entire theory and suspicions are incorrect. The investigator would then have to re-evaluate the evidence and consider alternative possibilities.

If the initial outlook was uncertain as to which evidence is to be trusted and which is to be disregarded, the model is only run until the trust relationships have been established according to the trust parameters in place. For instance, if a recommendation-based trust model is employed, the establishment of trust relationships relies on recommendations from trusted nodes. To recreate the environment as faithfully as possible, nodes that trust the suspect node are modified to trust the new virtual node to a similar degree. The investigator needs to be aware that sometimes the trust value may have to be rolled back to a different value that has since changed due to the effect of the criminal-related transactions on the trust environment itself. The degree to which the state of an environment can be rolled back depends on the prevailing trust model and requires further investigation.

Next, the trust relationship values between the virtual trust analyser node and the other nodes in the network are analysed. This is a fairly

simple concept as the trust relationships between nodes are often represented as single values. Nodes given a high trust value by the virtual trust analyser node are considered to be more trustworthy than those with lower values. Thus, the evidence contained in these nodes has a higher probability of being trustworthy. Trust models only allow a transaction to take place if the nodes participating in the transaction are trusted; otherwise, the transaction would not have taken place.

4. Discussion

The model proposed in this paper can be used by investigators to determine which evidence can be trusted and which evidence is suspect. Also, it can help recreate the crime and provide supporting evidence for the suspected crime. Note, however, that the model is preliminary in nature, and substantial research is required before it can be used in digital forensic investigations.

This model assumes that the network being examined for evidentiary purposes has certain trust mechanisms in place that control the interactions occurring in the context of the network. However, the model should also be applicable to networks that do not have explicit trust architectures in place. In such instances, the process of defining the trust parameters and trust environment will change. Instead of defining a trust model as in a network with trust mechanisms, a default trust context will have to be employed. Further research is necessary to define appropriate default contexts.

This model also assumes that the trust mechanisms work and have not been subverted by a criminal. It is necessary to examine how trust mechanisms might be subverted. The fact that trust models and their workings vary must be taken into account while researching this issue.

The reenacted transactions must be similar to those involved in the suspected crime. However, these transactions must be conducted carefully so that they do not modify data left by the original crime, but only add to it. Should the investigator find that the original data was altered during a reenactment, the transactions used to recreate the crime must be analysed and controlled more carefully. This is an interesting area for future research.

Substantial resources may be needed to conduct investigations on large networks. To reduce the complexity and investigative overhead, an investigator has the option of copying only critical portions of a large network and running the tool on those portions.

5. Conclusions

The trust-based forensic model proposed in this paper is intended to help evaluate forensic evidence to determine which evidence can be trusted and which evidence has been tampered with. This model also allows for crimes to be reenacted to create more substantial evidentiary proof.

The three phases of the model have been investigated from a conceptual point of view. More research is necessary to explicitly define what happens in each phase and how it should be accomplished. Areas that warrant attention are how the network may be copied to preserve the prevailing trust environment, how protocols will work on the network copy and how to explicitly define the crime activities being reenacted.

An interesting dilemma arises when a computerised agent is able to actively instigate transactions on behalf of a user. In such an environment, an agent is given rights to participate in transactions without the user's direct knowledge. Investigations must take into account the fact that criminal activity could have been caused by a code flaw or by a malicious act by the programmer of the agent code and not directly by the user. Methods for testing and proving code must be evaluated and incorporated in the proposed model.

Acknowledgements

This material is based on work supported by the National Research Foundation (NRF) under Grant No. 2054024. Any opinions, findings, conclusions and recommendations expressed in this material are those of the author(s) and, therefore, the NRF does not accept any liability thereto.

References

- [1] V. Baryamureeba and F. Tushabe, The enhanced digital investigation process model, presented at the *Digital Forensics Research Workshop*, 2004.
- [2] S. Bui, M. Enyeart and J. Luong, Issues in computer forensics (www.cse.scu.edu/~jholliday/COEN150sp03/projects/Forensic%20Investigation.pdf), 2003.
- [3] M. Carbone, M. Nielsen and V. Sassone, Gigascope: A formal model for trust in dynamic networks, *Proceedings of the First International Conference on Software Engineering and Formal Methods*, pp. 54-61, 2003.

- [4] M. Coetzee and J. Eloff, Towards web services access control, *Computers & Security*, vol. 23(7), pp. 559-570, 2004.
- [5] B. Esfandiari and S. Chandrasekharan, On how agents make friends: Mechanisms for trust acquisition, *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, pp. 27-34, 2001.
- [6] C. Jonker and J. Treur, Formal analysis of models for the dynamics of trust based on experiences, *Proceedings of the Ninth European Workshop on Modeling Autonomous Agents in a Multi-Agent World (Lecture Notes in Computer Science, Volume 1647)*, pp. 221-232, Springer, Berlin, Germany, 1999.
- [7] A. Josang, Prospectives for modeling trust in information security, *Proceedings of the Australasian Conference on Information Security and Privacy (Lecture Notes in Computer Science, Volume 1270)*, pp. 2-13, Springer, Berlin, Germany, 1997.
- [8] L. Kagal, T. Finin and A. Joshi, Trust-based security in pervasive computing environments, *IEEE Computer*, vol. 34(12), pp. 154-157, 2001.
- [9] M. Marx and J. Treur, Trust dynamics formalized in temporal logic, *Proceedings of the Third International Conference on Cognitive Science*, pp. 359-363, 2001.
- [10] R. McKemmish, What is forensic computing? *Trends and Issues in Crime and Criminal Justice*, no. 118, Australian Institute of Criminology, 1999.
- [11] B. Nooteboom, *Trust: Forms, Foundations, Functions, Failures and Figures*, Edward Elgar Publishing, Cheltenham, United Kingdom, 2002.
- [12] M. Patton and A. Josang, Technologies for trust in electronic commerce, *Electronic Commerce Research*, vol. 4, pp. 9-21, 2004.
- [13] S. Peron and M. Legary, Digital anti-forensics: Emerging trends in data transformation techniques, presented at the *E-Crime and Computer Evidence Conference*, 2005.
- [14] K. Ryder, Computer forensics: We've had an incident, who do we get to investigate? (www.sans.org/rr/incident/investigate.php), 2002.
- [15] A. Svensson, Computer Forensics Applied to Windows NTFS Computers, Master's Thesis, Stockholm University/Royal Institute of Technology, Stockholm, Sweden, 2005.
- [16] L. Xiong and L. Liu, A reputation-based trust model for peer-to-peer e-commerce communities, *Proceedings of the Fourth ACM Conference on E-Commerce*, pp. 228-229, 2003.

Chapter 6

EXPLORING BIG HAYSTACKS

Data Mining and Knowledge Management

Mark Pollitt and Anthony Whitledge

Abstract The proliferation of computer-generated evidence in court proceedings during the last fifteen years has given rise to the new science of digital forensics and a new breed of law enforcement officials, “computer forensic examiners,” who apply the rules of evidence, investigative methods and sophisticated technical skills to analyze digital data for use in court proceedings. This paper explores the technical challenges facing the law enforcement community and discusses the application of data mining and knowledge management techniques to cope with the increasingly massive data sets involved in digital forensic investigations.

Keywords: Digital forensic process, data mining, knowledge management

1. Introduction

The term “forensics” means “relating to, used in, or appropriate for courts of law or for public discussion or argumentation” [7]. In the judicial system, forensics refers to a branch of science, e.g., forensic pathology, devoted to providing data and conclusions as evidence in judicial proceedings. The goal of forensic procedures and protocols is to meet federal and state rules governing the admissibility and use of evidence in courts of law. For example, the Federal Rule of Evidence 901 [15] requires that information be “authenticated” to the court before it may be admitted into evidence. All forensic sciences have rigorous requirements designed to ensure that a trail exists backward from the scientific test to the original sample, and to prove that the scientific conclusion is about the sample in question.

Digital forensics has been defined as: “...the application of science and engineering to the legal problem of digital evidence” [12]. Digi-

Please use the following format when citing this chapter:

Pollitt, M., Whitledge, A., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 67–76.

tal forensics, then, is the science of collecting, preserving, examining, analyzing and presenting relevant digital evidence for use in judicial proceedings.

In the early days—around 1990—digital forensics focused almost exclusively on copying information from computers. With few tools available, examiners sometimes took days or weeks to retrieve pertinent information from a 5 MB or 10 MB hard drive. Usually, the examiner printed the information for the investigator, who could then use it in the same manner as any other piece of evidence. Most computer programs produced paper reports as their output, and the printouts did not seem to differ from other documents created by more traditional means.

During the past fifteen years, the availability of inexpensive computers and the explosive growth of the Internet have significantly altered how people communicate and record their transactions and interactions. The old paper-centric society has been transformed to a computer- or network-centric culture. Society has adapted the old ways of doing business to the new technologies and, at the same time, has found many new ways of utilizing them. Two enablers of this revolution are the dramatic increase in storage capacity and the equally dramatic decrease in the per-unit cost of storage.

Inexpensive storage and new technologies make it possible to create vast amounts of information quickly and easily, and eliminate the cost-based need to delete much of this information. Most investigations and legal cases focus on only a fraction of the information contained in a data set. Finding relevant information in a data set collected from an individual or business has become a search for the proverbial needle in a haystack. This paper discusses the inability of current forensic methodologies to deal effectively with the increasing volumes of data collected by investigators. In particular, it examines two approaches—data mining and knowledge management—that might provide the needed paradigm shift.

2. Data Volume

The FBI has reported dramatic increases in the number of cases involving digital evidence and the volume of digital evidence: from 2,084 cases and 17 terabytes of data in FY 1999 to 9,593 cases and 776 terabytes of data in FY 2004 [6] (see Figure 1). Data for FY 2005 indicate further increases in the number of cases and the volume of evidence.

This trend is not an FBI anomaly and is, in fact, representative of the entire digital forensics community. Researchers at the University of California, Berkeley have studied the amount of digital information

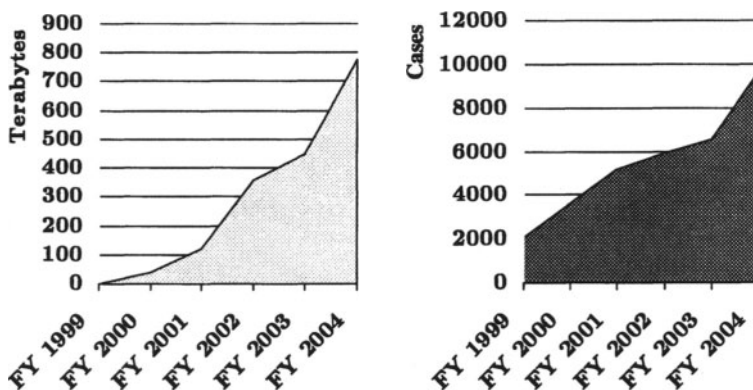


Figure 1. FBI-reported data and case volumes for FY 1999–2004 [6].

created annually [9]. By their estimates, the entire planet produced between 2 and 3 exabytes (10^{18} bytes) in 1999 and about 5 exabytes in 2002, corresponding to a growth of 30% per year. To put this volume of information in perspective, the Berkeley researchers equated the 5 exabytes to the equivalent of 37,000 Libraries of Congress, or more than 800 MB of data per living person. Incredibly, as much as 92% percent of the new information is stored on magnetic media, primarily hard disks [9]. Another source [16] has estimated that this volume of information represents “all the words ever spoken by human beings.” This trend is likely to continue for the foreseeable future.

Data volume, however, is not the only issue. If it were, simply adding computing resources would solve the problem. Rather, the increase in volume is indicative of increasingly massive data sets that hide small amounts of information relevant to an investigation. The goal of a digital forensic examination is to develop “information of probative value.” The determinants of success in this process are knowing what information to look for, discovering where to find it, and then recognizing it when it is encountered. However, existing methods, especially those that rely on the actions of a single forensic examiner or investigator, do not scale well and, therefore, do not adapt to large data sets.

3. Digital Forensic Process

Digital forensics has evolved over two decades. While the processes of investigation, examination and analysis of digital evidence have not been studied extensively, several papers address the general area [1, 3, 8, 10, 13]. While varying in their focus, all the papers recognize digital

forensics as part of a larger investigative process and one that contains definable functions. This view is consistent with the observations of the authors of this paper, each of whom supervised a national digital forensics program for a major U.S. law enforcement agency.

Over time, the digital forensic process has become somewhat standardized with slight variations based on organization, jurisdiction, purpose and available resources. The process has several components.

- *Media Acquisition:* Obtaining evidence by duplication or seizure.
- *Preservation/Duplication:* Ensuring the integrity of the evidence by a chain of custody and duplication of the original evidence.
- *Documentation:* Documenting the physical media, partitions, file systems and data.
- *Data Reduction:* Using known hash values and other techniques to eliminate data without probative value.
- *Data Selection:* Attempting to identify and extract probative information, e.g., using string searches, header examination, reconstruction of file system structures and files, content examination and the selection of data for further analysis.
- *Examination and Analysis:* Discovering and recording the technical provenance of data and its investigative context.
- *Reporting:* Creating written, oral and electronic products to communicate the results of the examination.

While digital forensic tools have become much more robust, the underlying process has changed little since the 1990s. As we discuss below, two characteristics of the digital forensic process appear to be limiting and both of them involve the need for human judgment.

The classic tasking from investigators to digital forensic examiners is: "*Tell me what is important that will help solve my case.*" This tasking makes two assumptions: the examiner knows what is important in the case, and the examiner can identify what is important. Both assumptions place substantial reliance on the examiner's ability to locate high value information and to recognize its significance. Success in this endeavor often boils down to the examiner's or investigator's ability to distill the volume of digital evidence into a manageable data set and to recognize important information when it is encountered. We would suggest that, at best, this process is becoming very difficult and, at worst, it is a gamble.

Thorp [14] has observed that IT-enabled change is creating dilemmas for management. So much information is being delivered by technology that many people now feel that they are drowning in information or are being forced to work with the wrong types of information. Thorp also emphasizes that while IT investment is producing more data, it is not providing the means to translate it into information and knowledge that have business value. This is a very apt description of the state of digital forensics: digital forensics has the ability to manage data, not information and knowledge.

4. Analyzing Large Data Sets

Current digital forensic practices are hardware-centric, focusing almost exclusively on the collection, extraction and presentation of data to investigators. What many forensic practitioners call “analysis,” is really part of the extraction process. The conversion of email messages to text-searchable files, the piecing together of data fragments from slack space and other areas of the disk, and the recovery of deleted files are processes that provide raw data to investigators. They are not analytical processes that help investigators understand the data or point them to the most important pieces of information.

Meanwhile, the increasing sophistication of operating systems and storage devices, and the massive volumes of stored data often result in forensic products that are too large and too sophisticated for their forensic examiner customers to understand or use effectively. Subject matter experts—agents and detectives—generally do not have strong computer backgrounds or skills. They may have little familiarity with data storage concepts, file system layouts, metadata and the relationships between the stored data and its content and meaning. Investigators also do not have the skills or equipment to handle the volumes of data being collected and provided to them: data volumes in the 500 GB to 1,000 GB range are increasingly common in fraud cases, and data volumes are on the rise in all types of cases.

On the other hand, the trend in law enforcement agencies is to move digital forensics into a laboratory model that imposes a rigorous scientific approach. This also separates forensic examiners from the consumers of their work product: agents, investigators and detectives who are responsible for cases. The result is a corps of forensics examiners who may not understand the subject matter of cases and are unable to provide adequate analytical assistance to consumers.

It is increasingly apparent that the task of analyzing case data comes after the forensic examiner’s work ends and before the agent’s investiga-

tive work begins. Forensic examiners generally do not have the analytical skills or the subject matter expertise to assist in data analysis. Indeed, neither forensic examiners nor investigators see substantive data analysis as part of their job descriptions. Given the complexity of analyzing terabytes of data and their background and training, it is unlikely that agents and investigators will master this vital part of the process at any time in the near future.

A notable exception is the Internal Revenue Service's Criminal Investigation Division (IRS-CID), which selects its digital forensic examiners from its cadre of experienced, senior agents. Its "Computer Investigative Specialists" (CISs) are thus both subject matter experts and forensic experts who often work closely with case agents to help with data analysis and "translate" the technical issues that affect the quality of the evidence. Although this works well for the IRS, it may not be adaptable to other law enforcement agencies. The IRS-CID is a specialized unit with jurisdiction over relatively few crimes; it is, therefore, easier for a CIS to be a subject matter expert on IRS casework than it would for a forensic examiner in a larger agency like the FBI. Additionally, many federal law enforcement agencies and police departments may not have the funding to devote investigative resources to digital forensics.

The remainder of this paper discusses strategies for dealing with the gap between forensic examination and investigative analysis in large cases. From the earliest days, law enforcement has adapted tools and techniques used for other purposes, e.g., systems administration and network analysis, to the digital forensic process. This paper proposes the next step: applying concepts and analytic tools designed for business data to the forensic process.

5. Data Mining

Data mining entails the discovery of meaningful patterns, rules or summaries in data (see, e.g., [2]). Data mining techniques cover both exploration and analysis. They may be automated or semi-automated, and are capable of dealing with large volumes of data.

Data mining is all about patterns and relationships between data elements. The use of data mining by businesses to understand and use transactional and other data is well established. Criminal investigators face the same challenges: to understand and use the data collected during investigations. Specifically, investigators must locate and understand the information that is relevant to their cases. It is equally important to find relevant information that the investigators did not know existed.

Investigators typically start with some knowledge and suspicions, and work to uncover facts that support their hypotheses. However, the massive volumes of case data make this an increasingly difficult task. Furthermore, patterns, connections and facts hidden in the data may not be discoverable without the assistance of analytical tools.

Current techniques for dealing with digital data range from merely printing out documents for subsequent analysis to the application of sophisticated data mining techniques. The techniques and tools used invariably depend on the importance of the case and the investigative dollars allocated to it. Thus, in smaller cases, an investigator is usually left to work with whatever hardware and tools are locally available. Large cases, or cases that attract media attention, are generally given greater resources and access to technically sophisticated analysts.

However, in most investigations, the analysis of digital data is limited to the manual examination of data files, string searches of data sets, and the use of databases to accumulate and sort pertinent information. While some forensic tools have scripting and programming capabilities, their use is normally limited to forensics experts, not the general investigative population. Consequently, automated tools are rarely, if ever, used to locate and analyze relevant evidence.

There are several reasons why investigators have limited experience with sophisticated analytical tools and data mining techniques and why these tools and techniques are scarcely used in investigations. In most law enforcement agencies, investigators and management have little or no understanding of the benefits data mining could bring to investigations. Moreover, agencies typically do not have funds for procuring tools and training investigators on the effective use of data mining and automated analytical techniques. Often, the cost and time required to convert digital evidence into a common format for analysis is a significant factor in the decision not to engage modern analytical techniques.

However, attempts to use advanced analytical tools in investigations have been successful, although not as successful as might be expected. The lead time to develop such tools dictates that generic approaches be adopted and such approaches may be too general to assist with specific cases. In other instances, investigators and developers expect too much from a tool. For example, visualization tools, e.g., the commercially-available Analysts Notebook, are effective at elucidating connections between related elements in a data set, but they do not help with data reduction or provide other types of analysis that might establish otherwise undiscoverable relationships.

As might be expected, the most successful results have come from cases where custom applications are tailored to a specific investigation

or data set. However, such applications are expensive and not adaptable to other types of cases and other data sets, although they do demonstrate their power and utility in investigations. It is expensive to convert collected case data to a common format needed for data mining and to develop the custom tools to mine the data effectively. Such an effort requires a cross-functional understanding of data mining operations and investigations. Yet, few in law enforcement understand data mining or the role it might play in investigations and even fewer data miners understand the goals, methods and requirements of criminal investigations.

In the authors' opinion, data mining tools can provide a means to bridge the gap between the forensic examiner's output and the investigator's quest for relevant information. It would be most useful, indeed, for law enforcement to have a set of tools designed to assist agents and analysts in developing patterns from data that may indicate criminal conduct, identifying individual items of relevant information from a large data set, and reducing data volume by eliminating redundancy and filtering irrelevant and unimportant information.

6. Knowledge Management

Knowledge management is "[a] discipline that promotes an integrated approach to identifying, capturing, evaluating, retrieving and sharing all of an enterprise's information assets. These assets include databases, documents, policies and procedures, and previously uncaptured tacit expertise and experience in individual workers" [11].

Davenport and Prusak [5] describe the relationships between three key elements involved in knowledge management: data, information and knowledge. Data is defined as "a set of discrete, objective facts about events." This accurately describes the portion of the digital forensic process that focuses on files, file systems and structures. A hard drive is thus interpreted as a massive number of data elements.

Davenport and Prusak describe information as "data that makes a difference" [5]. Data becomes information when the recipient recognizes value in the data. Information becomes knowledge when one or more of four processes occur: comparisons, consequences, connections and conversations. The conversions of data to information and information to knowledge are fundamental to knowledge management, and involve the application of human intellect [5].

Thorp [14] describes knowledge management in terms of a "benefits realization approach." He proposes two techniques: modeling and value assessment. The former technique uses a "results chain" method, where outcomes are modeled against assumptions, initiatives and contri-

butions. The latter incorporates four criteria to evaluate the conversions: alignment, benefits, integration and capability/efficiency.

Another key aspect of knowledge management is the conversion of tacit information to implicit information. In the forensic and investigative context, this is clearly a significant issue, as it frames the inquiry and either contributes to or detracts from the outcome [4]. Currently, the conversion of tacit information between examiners/investigators and attorneys is done on an *ad hoc* basis, without any formal structure.

Knowledge management can significantly enhance the practice of digital forensics. Three areas for future research are modeling; data, information and knowledge valuation; and developing effective strategies and methodologies for conversion of tacit information to explicit knowledge. Failure to develop effective mechanisms will likely impede the effectiveness and utility of digital evidence.

7. Conclusions

Criminal investigators at all levels have to deal with increasingly complex cases with massive amounts of digital evidence. Meanwhile, the gap between digital forensics experts, who collect, secure and present digital evidence, and investigators who use digital evidence, is widening. The outputs of forensic examiners are often too large and too complex for investigators to use effectively. The law enforcement community needs assistance in managing large data sets. Also, it needs analytical tools that can work across the disparate data types in investigative data sets. These tools should be easy to use and should produce results that the investigators can understand and explain in court.

Data mining tools and knowledge management principles can (and should) be developed to help address these issues. In particular, research should be directed at developing knowledge management strategies specific to law enforcement that will operate within the specific context of criminal investigations. Furthermore, research should focus on forensic applications of data mining tools and developing analytic tools for forensic examiners and criminal investigators.

References

- [1] N. Beebe and J. Clark, A hierarchical, objectives-based framework for the digital investigation process, *Proceedings of the Digital Forensics Research Workshop*, 2004.
- [2] M. Berry and G. Linoff, *Data Mining Techniques for Marketing, Sales and Customer Support*, John Wiley, New York, 1997.

- [3] B. Carrier, An event-based digital forensic investigation framework, presented at the *Digital Forensics Research Workshop*, 2004.
- [4] B. Crowley, Tacit knowledge and quality assurance: Bridging the theory-practice divide, in *Knowledge Management for the Information Professional*, K. Srikantaiah, M. Koenig and T. Srikantaiah (Eds.), Information Today, Medford, New Jersey, 2000.
- [5] T. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press, Boston, Massachusetts, 1998.
- [6] A. DiClemente, Digital forensics: Current status and future directions, presented at the *First IFIP WG 11.9 International Conference on Digital Forensics*, 2005.
- [7] Farlex, Inc., The Free Dictionary (www.thefreedictionary.com).
- [8] G. Hama and M. Pollitt, Data reduction – Refining the sieve, presented at the *Second International Conference on Computer Evidence* (www.digitalevidencepro.com/Resources/Sieve1.pdf), 1996.
- [9] P. Lyman and H. Varian, How Much Information 2003? (www.sims.berkeley.edu/how-much-info-2003), 2003.
- [10] M. Pollitt, A framework for digital forensic science, presented at the *Digital Forensics Research Workshop*, 2004.
- [11] The Provider's Edge, LLC., Knowledge Management Basics (www.providersedge.com/kma/km_overview_km_basics.htm), 2003.
- [12] A. Sammes and B. Jenkinson, *Forensic Computing: A Practitioner's Guide*, Springer-Verlag, New York, 2000.
- [13] P. Stephenson, Modeling of post-incident root cause analysis, *International Journal of Digital Evidence*, vol. 2(2), 2003.
- [14] J. Thorp, *The Information Paradox: Realizing the Business Benefits of Information Technology*, McGraw-Hill, Toronto, Canada, 1999.
- [15] U.S. Government, *Federal Rules of Evidence* (judiciary.house.gov/media/pdfs/printers/108th/evid2004.pdf), 2004.
- [16] R. Williams, Data Powers of Ten (www.davedoyle.com/help/data.html), 2005.

Chapter 7

COUNTERING HOSTILE FORENSIC TECHNIQUES

Scott Piper, Mark Davis and Sujeet Sheno

Abstract Digital forensic investigations can be subverted by hostile forensic techniques and tools. This paper examines various hostile forensic techniques, including the exploitation of vulnerabilities in standard forensic procedures and denial of service attacks on forensic tools during imaging and analysis. Several techniques for concealing evidence within file systems and external to file systems are highlighted. In addition, strategies for countering hostile forensic techniques and tools are discussed.

Keywords: Hostile forensics, subversion, denial of service, evidence concealment

1. Introduction

The emerging discipline of digital forensics brings advanced scientific and engineering techniques to bear on the tasks of detecting, recovering and analyzing electronic evidence [12, 14]. However, certain elements of the hacker community are engaged in developing “anti-forensic” or “hostile forensic” techniques and tools to subvert digital forensic investigations [8, 13, 15]. Efforts have been undertaken to exploit flaws in digital forensic techniques and tools. The holy grail of these efforts is to find an exploit, e.g., a buffer overflow, that would result in the execution of malicious code in forensic tools used by law enforcement agencies. Such an exploit could make the tools unable to display certain data, make them delete evidence, or simply prevent them from operating. Fortunately, such an exploit has not yet been created, but hostile forensic techniques and tools abound [8, 13, 15].

Some hostile forensic techniques hinder investigations by hiding evidence, destroying evidence or by ensuring that little or no evidence is created. Others exploit vulnerabilities in forensic procedures and tools

Please use the following format when citing this chapter:

Piper, S., Davis, M., Sheno, S., 2006 in International Federation for Information Processing, Volume 222. Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 79–90.

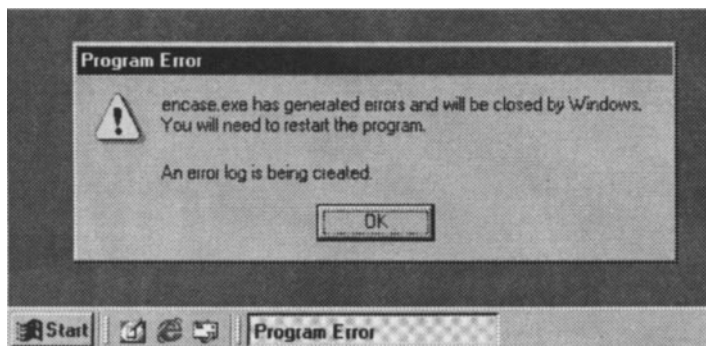


Figure 1. Result of a denial of service attack on EnCase v4.15.

to prevent evidence from being discovered. More insidious are hostile techniques that perpetrate denial of service attacks on forensic tools to prevent them from imaging media or analyzing evidence.

Nothing can be done when no digital evidence is created. When digital evidence is destroyed, not much can be done to recover it without the use of specialized equipment. However, if evidence does exist on a computer system or network, it can be found and analyzed, even when hostile forensic techniques have been employed.

This paper discusses the state of the art in hostile forensics and presents strategies for countering hostile forensic techniques and tools. The next section focuses on techniques for subverting investigations and perpetrating denial of service attacks on forensic tools. Following this, techniques for concealing evidence within file systems are discussed. Finally, strategies for hiding data in devices that are not normally seized and in devices that are not easily imaged are evaluated.

2. Subversion and Denial of Service

Investigations cannot proceed if the forensic tools themselves are incapable of detecting, recovering or analyzing evidence from computer systems. One hostile forensic strategy is to exploit a vulnerability in a forensic technique or procedure to prevent the discovery of evidence. Another is to launch a denial of service attack on a forensic tool during imaging or analysis to simply prevent it from operating properly.

Many file systems permit the creation of arbitrarily deep directory structures. However, EnCase v4.15 is not designed to traverse a directory tree more than 253 directories deep. When an investigator attempts to use EnCase v4.15 to examine such a directory tree, the tool exhibits a fatal error (Figure 1).

This section discusses techniques for defeating forensic procedures and tools. These techniques, involving the manipulation of timestamps, insertion of compression bombs, and use of sparse files and magic numbers, are described along with strategies for countering them.

2.1 Timestamps

A computer incident, e.g., a network intrusion, occurs within some period of time. Forensic investigators focus the majority of their efforts on discovering what happened during that time frame. The analysis typically involves reviewing the modification, access and creation (MAC) times of files to determine what data may have been accessed during the intrusion. Many forensic tools provide functionality for filtering files that meet the temporal criteria pertaining to an incident. Malicious individuals have attempted to subvert investigations by changing the timestamps associated with incriminating files.

The Linux `touch` command can be used to change file timestamps to the current time, and to reset MAC times to arbitrary values. Several utilities (e.g., `fileTweak`) are available for changing timestamps on Linux, FAT and NTFS file systems. The Metasploit Project [13], which produced a framework for developing exploit code, created the `Timestamp` utility that targets NTFS files. In addition to MAC times, NTFS files have an “entry modified” time. `Timestamp` is the first tool that permits the modification of all four timestamps associated with NTFS files [13].

File timestamps cannot be trusted. This fact should be taken into account when filtering files during the examination of evidence recovered from a computer system or network. Other evidence, such as access logs, should be considered when searching for files that are relevant to an incident.

2.2 Compression Bombs

A compression bomb is a file that expands massively when it is uncompressed, causing the system to crash [2]. For example, the `42.zip` [1] compression bomb is a mere 42KB. When uncompressed completely, it expands to an astounding 4.5PB (Petabytes).

The technique for creating a compression bomb is relatively simple. First, a large file containing zeroes is compressed (Figure 2). Multiple copies of this compressed file are combined into a new file, which is compressed. Copies of the new compressed file are then made, and the copies are compressed again into a single file. This procedure is repeated to produce a small – but extremely potent – compression bomb.

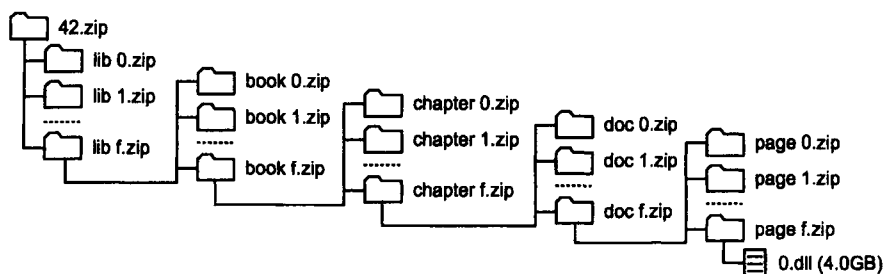


Figure 2. Uncompressed version of the 42.zip compression bomb.

Compression bombs were originally used to disable anti-virus filters. Typically, an attacker would email a compression bomb to a targeted computer system. The anti-virus software on the email server would attempt to scan the compression bomb for viruses by uncompressing it. In the process, the anti-virus software would exhaust its memory resources and crash, exposing an open portal for the attacker to send the real virus. Similarly, an individual wishing to disrupt an investigation might store several compression bombs on a hard drive or other computer media, causing digital forensic tools to crash.

The commonly used digital forensic tools react differently to compression bombs. The Forensic Toolkit (FTK) (v1.42 build 03.12.05) freezes and becomes unusable when it attempts to acquire the image of a drive containing a compression bomb. EnCase (v4.15) is able to acquire an image, but it freezes when it burrows too deep into the compression tree. ILook (v7.0.35), on the other hand, is unable to traverse more than one compression level; therefore, a file that is compressed two or more times is inaccessible to ILook.

Once a compression bomb has been identified, it can be ignored for the purpose of gathering evidence. High compression ratios are achieved by compressing files composed entirely of zeroes. Consequently, it is unlikely that useful data is stored within a compression bomb.

2.3 Sparse Files

Sparse files are an obscure feature of Ext2/Ext3 and other file systems (e.g., NTFS) [5]. These files allow data to be written to any location within the file. A sparse file has few data items, and the locations that do not hold data are assigned zero values. It is inefficient to have many blocks on a disk that contain identical data values, especially when the values are all zeroes. Consequently, sparse files map all blocks with zero values to a single block on the disk (e.g., Block 0 for Ext2/Ext3).

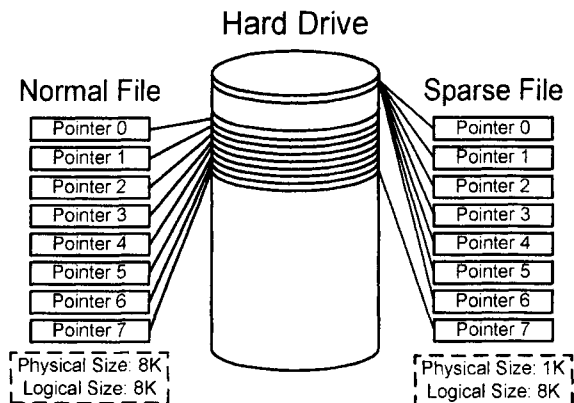


Figure 3. Comparison of normal file and sparse file storage.

Figure 3 shows how normal and sparse files are stored in a file system with 1K blocks. Pointers 0 through 6 of the sparse file (corresponding to blocks with zero values) point to the same block. Consequently, while the logical size of the sparse file is 8K, its physical size is only 1K.

```
# echo "This is a 1GB file" | dd of=sparse.txt bs=1K seek=1048576
0+1 records in
0+1 records out

// Now show the file sizes; logical, then physical
# ls sparse.txt
-rw-r--r-- 1 root    root      1.0G Jun  3 16:19 sparse.txt
# du sparse.txt -h
4.0K    sparse.txt
```

Figure 4. Sparse file creation procedure.

Figure 4 presents a Linux console procedure for creating a 1GB sparse file on an Ext2 formatted floppy disk. A 1TB sparse file can also be created on a floppy disk. However, when formatting the disk, it is necessary to set the number of inodes to a value greater than the default value.

Digital forensic tools react to sparse files in unpredictable ways depending on the size of the file. FTK breaks a 1GB sparse file into smaller files; all the contents are stored in the last file, which may be viewed using FTK. On the other hand, while FTK will show that a 1TB sparse file exists, it cannot display its contents. EnCase crashes when used to view a 1GB sparse file. Like FTK, EnCase will show that a 1TB sparse file exists, but it cannot display its contents. Regardless of its size, ILook will show that a sparse file exists but it cannot display its contents.

Note that a sparse file may contain important evidence anywhere within the file, not just at the end. Therefore, after determining that a sparse file exists, an investigator must review a hex dump of the file system to analyze the file.

2.4 Magic Numbers

Every file system has a signature, called a “magic number,” that allows the operating system to determine its format [5]. The Ext2/Ext3 file system has a 2-byte magic number of 53 EF; FAT has a magic number of 55 AA. File systems continue to function normally even when the magic number is corrupted. However, most software, including digital forensic tools, cannot determine the correct file system, which prevents them from functioning properly. For example, a forensic tool would not be able to parse the data structures in an imaged file, although it would still permit hex dumps of the image.

The magic number on an Ext2 formatted floppy disk can be overwritten by issuing the Linux command:

```
# dd if=/dev/zero of=/dev/fd0 bs=1 count=2 seek=1080.
```

If the magic number (or the beginning) of a partition is overwritten, it is first necessary to determine the file system. Next, the magic number in the image must be corrected to permit analysis using forensic tools.

A related denial of service attack involves overwriting a partition table so that forensic tools cannot determine where partitions begin and end. The tools assume that the entire drive is one giant partition and do not parse the real partitions correctly. The Linux utility **gpart** [3] can be used to reconstruct the partition table in such a situation.

3. Data Concealment within File Systems

This section describes how secret information may be hidden within file systems to evade detection by traditional digital forensic tools. Three data hiding techniques, involving alternate data streams, file slack space and reserved locations of file systems, are discussed along with strategies for detecting and recovering hidden data.

3.1 Alternate Data Streams

Microsoft Windows is commonly installed on a hard disk using the NTFS file system. NTFS provides more functionality than FAT, which was used in earlier versions of Windows. Alternate data streams, one of the new features of NTFS, can be used to conceal data [12].

NTFS files are interpreted by the operating system as streams of data associated with a filename. Typically, files only have one stream of data, but additional data streams may be added to store information about the file, e.g., summary information about the file, keywords and comments. Within the Windows XP GUI, an alternate data stream can be created or viewed by right-clicking on a file, selecting **Properties** and then selecting the **Summary** tab.

```
C:\>echo hello world > file.txt
C:\>echo this data is hidden > file.txt:secret
C:\>more < file.txt
hello world
C:\>more < file.txt:secret
this data is hidden
```

Figure 5. Alternate data stream creation procedure.

The DOS command prompt can also be used to create alternate data streams. Alternate data streams are created in the same way as normal files, but the file is referenced using the file name and the alternate data stream name separated by a colon.

Figure 5 shows the procedure for creating a file with contents **hello world**. An alternate data stream called **secret** is associated with the file; this alternate data stream contains **this data is hidden**. Figure 5 also shows that when file contents are displayed, only the data associated with the default stream (**hello world**) appears. The alternate data stream is displayed using the command: **more < file.txt:secret**. This demonstrates that the contents of the alternate data stream are distinct from the default stream.

Alternate data streams are useful for concealing data because the Windows operating system lacks the functionality to access them. Indeed, Windows ignores alternate data streams when reporting file sizes and free space on a disk. For example, the file **file.txt**, which contains both **hello world** and **this data is hidden**, is listed as being only 14 bytes. Alternate data streams are not listed when viewing directory listings or browsing folders using Windows Explorer. In fact, the only way to discover an alternate data stream is to use third-party software, e.g., **Streams** [19].

Even more astonishing is the fact that, in Windows, the only way to delete an alternate data stream is to delete the entire file. Since alternate data streams can be associated with files as well as directories (including the root directory), the removal of an alternate data stream is somewhat

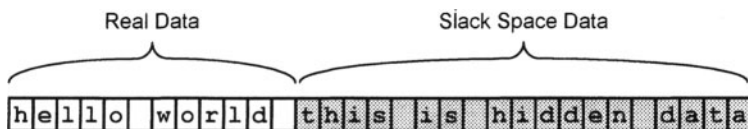


Figure 6. Hidden data in file slack space.

problematic. The **Streams** utility [19] can be used to selectively delete alternate data streams.

Until a few years ago, data concealed using alternate data streams could be discovered only by string searches and hex dump analyses. Most forensic tools are now able to detect the presence of alternate data streams.

3.2 Slack Space

Most file systems divide their partitions into blocks of equal size. Instead of allocating just enough bytes to store a particular file, complete blocks are reserved for the file. For example, on a file system with 512-byte blocks, a 14-byte file takes up 512 bytes of storage, and a 526-byte file uses 1024 bytes. Thus, files can grow within their allocated blocks; when they outgrow them, additional blocks can be allocated from elsewhere on the disk. Slack space is the unused space within a block [5, 12].

File slack space is not overwritten unless the size of the file increases. If the file shrinks, old data residing in the slack space could be retained indefinitely.

Data may be hidden in slack space (Figure 6), for example, by using the **bmap** tool that was originally created to read slack space. Many files, especially those associated with the operating system and applications, are updated rarely, if ever. The slack space of these files is a good place to hide data. Most forensic tools can be used to examine slack space, but investigators must know which files are most appropriate for hiding data and search their slack space for concealed evidence.

3.3 Reserved Locations

File systems have reserved locations that are used to support upgrades and new features. Since the reserved locations are unused until a file system is updated, data written to these locations neither overwrites useful data nor affects system operation.

The reserved locations of the Ext2/Ext3 file systems can be identified by reviewing Linux kernel source code in `./include/linux/ext2_fs.h`. Figure 7 shows the source code for one of the structures in the Ext2 file

```
struct ext2_group_desc
{
    __u32    bg_block_bitmap;        /* Blocks bitmap block */
    __u32    bg_inode_bitmap;        /* Inodes bitmap block */
    __u32    bg_inode_table;         /* Inodes table block */
    __u16    bg_free_blocks_count;    /* Free blocks count */
    __u16    bg_free_inodes_count;    /* Free inodes count */
    __u16    bg_used_dirs_count;      /* Directories count */
    __u16    bg_pad;
    __u32    bg_reserved[3];
};
```

Figure 7. Source code for the group descriptor.

system. A total of 14 bytes of data can be hidden within this structure, 2 bytes in **bg_pad** and 12 bytes in the **bg_reserved** variable.

The Data Mule FS tool [8] was designed to hide data in Ext2/Ext3 file systems. This tool breaks up a large file into small fragments, which are stored in reserved locations throughout a file system. To counter this tool, we developed the **rfinder** utility [17] that detects and extracts hidden data in Ext2/Ext3 file systems in a forensically sound manner.

4. Data Concealment outside File Systems

Standard forensic procedures involve seizing and imaging storage media. Individuals seeking to conceal evidence may hide data in devices that are not normally seized or in devices that are not easily imaged. This section describes how data may be concealed within random access memory, obscure hard drive locations and BIOS chips. Also, techniques for detecting and recovering hidden data are discussed.

4.1 Random Access Memory

The question of whether or not a running computer should be turned off upon seizure is a subject of debate [12, 14]. One side recommends pulling out the power cord. Another side, concerned that this procedure may damage the drive or stop the machine from completing a write operation, insists that the machine be shut down properly using the operating system. Yet another side, recognizing that valuable data might be lost during machine shutdown, recommends that information pertaining to open ports, running processes, etc. be collected while the machine is running. Each procedure has its advantages and disadvantages. However, the third procedure is based on an important observation – some key evidence may not be stored on disk.

To reduce the amount of evidence potentially recoverable from a hard drive, a malicious individual might attempt to perform most, if not all, actions in memory. A remote user could use a root kit that remains persistent in memory, and attach to a currently running process or use common utilities already on the machine to perform actions. Individuals desiring to minimize evidence of their actions on a local machine could use Knoppix [11] or other CD-bootable operating systems that do not require a hard drive or other permanent memory storage. The Tinfoil Hat Linux operating system is designed to leave no evidence pertaining to user actions: it encrypts all data written to persistent memory.

4.2 Hard Drives

A hard drive has more memory than is accessible by imaging the drive. For example, the Host Protected Area or ATA-Protected Area at the end of a hard drive cannot be read from or written to using standard operating system calls because the drive reports that it is smaller than its true capacity [5]. Forensic examiners should be aware that important evidence might be concealed in these locations. While standard tools (FTK, EnCase and ILook) cannot access this evidence, special tools, e.g., X-Ways Replica [20], are capable of detecting and recovering the hidden data.

SMART technology [16], another obscure hard drive feature, could be used by a remote hacker to determine if a victim machine has been the subject of an investigation. This technology, which is used to monitor the health of hard drives, provides information about how long a drive has been in operation. Suppose a hacker loses a connection to a victim computer for a period of time. Upon regaining the connection, the hacker could determine that the length of time that the drive has been in operation does not match the time elapsed since it was mounted. The hacker might infer that drive was imaged, and then attempt to subvert the investigation by wiping incriminating evidence on other computers.

4.3 BIOS Chips

Every computer and embedded device has a Basic Input/Output System (BIOS) chip, which is required to boot the system. A BIOS chip typically has 128K to 512K of flash memory that holds code and data. However, the chip may contain between 25K to 100K of unused space that can be used to store data without affecting the operation of the BIOS. This unused space has been exploited by virus writers and computer game enthusiasts. Malicious individuals can also use this space to hide incriminating evidence [6, 7].

Uniflash [18], a BIOS flashing utility, can be used to read and write data to a BIOS chip. Data may be written to BIOS free space and certain regions of BIOS modules (e.g., those containing error messages) without corrupting the BIOS [6, 7]. Alternatively, the entire BIOS memory may be overwritten, which, of course, renders the BIOS chip unusable [6, 7]. In this case, however, a BIOS Savior device [10] is required to boot the computer. This device provides a backup BIOS chip and a hardware switch that enables the user to select whether the computer will use the backup chip or the original BIOS chip for the booting process.

Forensic investigators must be aware that data may be hidden on a BIOS chip. They should check for utilities (e.g., Uniflash) and tools (e.g., BIOS Savior) that enable BIOS chips to be modified. It may also be necessary to conduct a forensic examination of the BIOS chip itself. Certain segments of BIOS memory can be viewed using the Windows **debug** command [6, 7]. The entire BIOS memory can be extracted using special software (e.g., AwardMod [9]) and analyzed using standard forensic tools (e.g., EnCase, FTK and ILook) [6, 7].

5. Conclusions

As digital evidence becomes increasingly important in judicial proceedings, it is logical to assume that malicious individuals will attempt to subvert investigations by targeting vulnerabilities in digital forensic procedures and tools. They will also endeavor to conceal incriminating evidence in obscure regions of file systems, in devices that are not easily imaged or in devices that are not normally seized.

This paper has two main contributions. The first is the description of the state of the art in hostile forensic techniques and tools. The second, and more important, contribution is the discussion of strategies for countering hostile forensic techniques and tools. Of particular significance are the strategies for combating subversion and denial of service attacks on forensic tools, and techniques for detecting and extracting concealed evidence. This paper has been written to raise awareness about hostile forensic techniques – and countermeasures – within the law enforcement community. We hope it will stimulate efforts within the digital forensics research and development community to ensure that all the evidence – wherever it may reside – is recoverable and presentable in court.

References

- [1] 42.zip (www.unforgettable.dk).
- [2] AERASec, Decompression bomb vulnerabilities (www.aerasesec.de/security/advisories/decompression-bomb-vulnerability.html).

- [3] M. Brzitzwa, **gpart** (www.stud.uni-hannover.de/user/76201/gpart).
- [4] R. Card, Cross-referencing Linux (lxr.linux.no/source/include/linux/ext2_fs.h?v=2.6.10).
- [5] B. Carrier, *File System Forensic Analysis*, Addison-Wesley, Crawfordsville, Indiana, 2005.
- [6] 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, New York, pp. 217-230, 2005.
- [7] P. Gershteyn, M. Davis and S. Shenoi, Forensic Analysis of BIOS chips, in *Advances in Digital Forensics II*, M. Olivier and S. Shenoi (Eds.), Springer, New York, pp. 301-314, 2006.
- [8] The grugq, The art of defiling, presented at the *Hack in the Box Conference* (packetstormsecurity.nl/hitb04/hitb04-grugq.pdf), October 8, 2004.
- [9] J. Hill, AwardMod (sourceforge.net/projects/awardmod), 2002.
- [10] IOSS, RD1 BIOS Savior (www.iooss.com.tw), 2000.
- [11] Knoppix (www.knoppix.org).
- [12] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*, Addison-Wesley, Boston, Massachusetts, 2002.
- [13] V. Liu, Metasploit Project (www.metasploit.com/projects/antiforensics).
- [14] K. Mandia and C. Proise, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.
- [15] S. McClure, J. Scambray and G. Kurtz, *Hacking Exposed: Network Security Secrets and Solutions*, McGraw-Hill/Osborne, Emeryville, California, 2001.
- [16] S. McLeod, Smart anti-forensics (www.forensicfocus.com/index.php?name=Content&pid=53).
- [17] S. Piper, M. Davis, G. Manes and S. Shenoi, Detecting misuse in reserved portions of Ext2/3 file systems, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, New York, pp. 245-256, 2005.
- [18] Rainbow Software, Uniflash (www.uniflash.org), 2005.
- [19] M. Russinovich, Streams (www.sysinternals.com/Utilities/Streams.html), 2005.
- [20] X-Ways Software Technology, X-Ways Replica: DOS disk cloning and imaging tool (www.x-ways.net/replica.html).

Chapter 8

USING PLSI-U TO DETECT INSIDER THREATS FROM EMAIL TRAFFIC*

James Okolica, Gilbert Peterson and Robert Mills

Abstract Despite a technology bias that focuses on external electronic threats, insiders pose the greatest threat to commercial and government organizations. Once information on a specific topic has gone missing, being able to quickly determine who has shown an interest in that topic can allow investigators to focus their attention. Even more promising is when individuals can be found who have an interest in the topic but who have never communicated that interest within the organization. An employee's interests can be discerned by data mining corporate email correspondence. These interests can be used to construct social networks that graphically expose investigative leads. This paper describes the use of Probabilistic Latent Semantic Indexing (PLSI) [4] extended to include users (PLSI-U) to determine topics that are of interest to employees from their email activity. It then applies PLSI-U to the Enron email corpus and finds a small number of employees (0.02%) who appear to have had clandestine interests.

Keywords: Probabilistic Latent Semantic Indexing (PLSI), insider threat, data mining, social networks

1. Introduction

"Espionage is the practice of spying or using spies to obtain information about the plans and activities of a foreign government or a competing company" [9]. While professional spies can be inserted into an organization, today, the use of insiders is much more prevalent. Insiders are members of an organization who often have a legitimate right to the

*The views expressed in this article 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.

information that they are accessing. However, they abrogate the trust they have been given by using the information for illegitimate reasons.

The best time to address the insider threat is before it occurs, i.e., prevention. The next best time is as soon as it is detected and before the perpetrator can cover his tracks. Given the scarcity of time and the large number of employees, investigators must quickly winnow the number of suspects to a manageable number. As more and more information is stored electronically, it is now possible to mine electronic information and extract likely investigative leads. One of the best indicators of an employee's interests is email. Through data mining, topics of interest can be extracted from email and people can be categorized by the topics they are most interested in. People who have shown an interest in sensitive topics can be investigated further. Likely suspects are individuals who have shown an interest in those topics but have never communicated that interest to anyone within the organization.

In this paper, Probabilistic Latent Semantic Indexing (PLSI) [4] is expanded to include users and then used on the Enron email corpus to test its applicability to generating investigative leads. The resulting PLSI-U (PLSI with users) model performs well, creating 48 clear categories and extracting 103 employees with clandestine interests. Given that only 103 individuals emerged from a population of more than 34,000 employees, the algorithm appears to produce a manageable number of investigative leads.

2. Motivation

At a RAND workshop on the insider threat [3], the first priority for improving the detection of insider's misuse was "developing [user] profiling as a technique" [10]. One way to detect potential insiders is to consider whether a person's interests match with the people with whom they are in contact. A profile describing a person's interests is generated by analyzing the content of their email. If someone shows a high degree of interest in a specific topic but does not email anyone else within the organization who also has an interest in that topic, it may suggest a clandestine interest. If the category is also relevant to an insider threat investigation, the person may warrant additional attention.

Electronic mail is fast becoming the most common form of communication. In 2006, email traffic is expected to exceed 60 billion messages daily [6]. While using email as data is not new, it has only recently begun to emerge as a tool for detecting deceptive communications [5]. Semantic analysis has been directly applied to countering insider threats by Symonenko, *et al.* [12]. They investigated the effectiveness of us-

ing natural language processing (NLP) to discover intelligence analysts who were accessing information outside of their community of interest. By using interviews with analysts to acquire significant domain specific knowledge, the researchers were able to use clustering to determine when an analyst was looking at (or producing) reports on areas other than the ones assigned to his group. While their success is impressive, it requires a significant amount of work to develop the domain specific knowledge. Furthermore, once this knowledge is acquired, the resulting model is only applicable to one domain. In contrast, the model described in this paper works without any domain knowledge and in a more generalized setting.

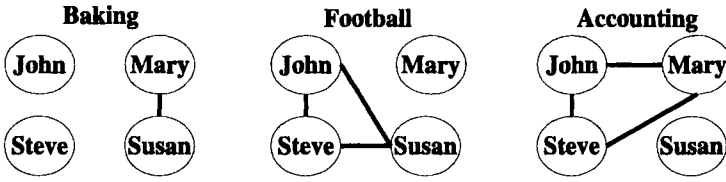
3. Methodology

This paper examines the potential use of constructing social networks from email activity to generate insider threat leads. The first step is developing user “interest profiles.” These profiles are generated through probabilistic clustering algorithms derived from the PLSI-U model. The profiles are then used to generate an implicit social network between people for each interest. Individuals are considered connected if they share an interest. A second explicit social network for each interest is then constructed based on email activity (containing that interest) between individuals. These two networks are compared for discrepancies. People who fail to communicate via email for a specific interest (i.e., not connected to anyone according to the explicit social network) but who have shown an interest (i.e., connected according to an implicit social network) are considered as possibly having a clandestine connection and are worthy of additional investigation.

Consider the example in Figure 1. Susan’s email correspondence indicates that she has an interest in football. However, none of the emails she sent or received *within the company* involve football. Therefore, for Susan, football is a clandestine interest. By varying the subset of interests that generate the networks (e.g., limiting it to suspicious interests), these clandestine connections become more relevant.

The first step is to use PLSI-U to cluster email activity into relevant group interests, or topics. Once the data has been clustered, building the social networks is straightforward. First, an implicit network is constructed from the PLSI-U data. If two people have an interest in a topic that exceeds a threshold, specifically 0.5%, a link is created between those two people. Mathematically, if $p(z = Z_1 | u = U_1) > 0.005$ and $p(z = Z_1 | u = U_2) > 0.005$, where Z_1 is a topic and U_1 and U_2 are people, then the link $U_1 U_2$ is created for the implicit PLSI network for

Implicit Interest Networks



Explicit Interest Networks

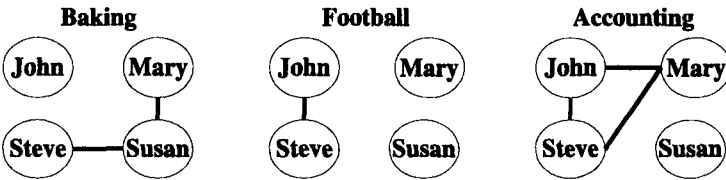


Figure 1. An example of clandestine interests.

category Z_1 . This process is repeated for every pair of people for every topic.

Once the implicit network is formed, an explicit network is created based on email data. If there is at least one email message for a specific topic between two people, a link is created between them. Mathematically, if $p(z = Z_1 | d = D_1) > 0.005$, where D_1 is an email, then $\forall U_1 \in D_1 \forall U_2 \in D_1$ the link $U_1 U_2$ is created for the explicit network for category Z_1 . This process is repeated for every topic and every pair of people.

The final step is to examine the implicit and explicit social networks. Each implicit network is compared to the explicit network in turn. If a person has an interest in a topic (i.e., there are links between that person and others in the implicit network) but has no links to anyone in the explicit network for that topic, an exception is generated.

4. Generative Model

This section describes the theoretical background used to develop the statistical model for predicting the likelihood that a specific email is constructed from a specific topic, and consequently is a member of a particular topic.

Notationally, M is the number of emails, $d_{i=1..M}$, in the corpus. There are V words in the vocabulary and each email, d_i , is composed of N_i

words, $w_{j=1..N_i}$. Furthermore, there are K topics. For simplicity, each email is considered to have a non-zero probability of each topic, $z_{r=1..K}$. Finally, each email has exactly one sender and one or more recipients. In this paper, the roles of these people are not distinguished (see [7] for models where roles are distinguished) and so each email, d_i , is considered to have L_i people, $u_{s=1..L_i}$, associated with it, drawn from a population of P individuals.

For simplicity, we use the naive Bayes assumption that each topic in an email is conditionally independent of every other topic and that every word and person are conditionally independent of every other word and person conditioned on the topic. Although this assumption is obviously wrong (e.g., “the cat ate the mouse” is different than “the mouse ate the cat”), techniques that make this assumption still produce good results.

PLSI is a generative model for the creation of a document within a corpus. However, it does not include the concept of people. Therefore, to use PLSI as a generative model for email, the concept of people requires incorporation, generating a new model, PLSI with users (or PLSI-U). PLSI-U assumes an email is constructed by first adding a user at a time and then adding a word at a time. Before each word or user is added, a topic is selected from a multinomial distribution and then the word or user is selected conditionally given the topic from a multinomial distribution. What is most desired is the joint probability of a word w_i and user u_s occurring in email d_j that contains topic z_r . However, given the size of the vocabulary, the number of people in the population, the number of words and people in the email corpus and the number of topics, determining the full joint probability is unrealistic. However, it is sufficient to determine the probability of topic z_r for a specific email. Then, by examining the probabilities for all of the topics, one can determine which topics the email contains (since they have the greatest probabilities). Therefore, the goal is to determine $p(z_r|d_j)$. However, given the generative model, there is no direct relationship between topics and emails. A topic “produces” words and the collection of words creates the emails. Therefore, in order to determine $p(z|d)$, it is first necessary to consider $p(z|d, w, u)$.

To begin, using Bayes rule:

$$p(z|u, d, w)p(u, d, w) = p(u, d, w|z)p(z). \quad (1)$$

Now, consider the model in Figure 2 and observe that u , d , and w are all conditionally independent given z . Therefore,

$$p(z|u, d, w) = \frac{p(u|z)p(d|z)p(w|z)p(z)}{p(u, d, w)}. \quad (2)$$

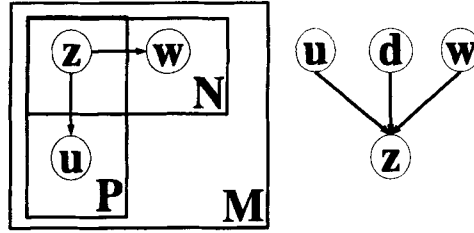


Figure 2. PLSI-U mixture model.

But $p(u, d, w)$ is simply $p(u, d, w|z)$ marginalized across all possible z values. So finally,

$$p(z|u, d, w) = \frac{p(u|z)p(d|z)p(w|z)p(z)}{\sum_{z' \in Z} p(u|z')p(d|z')p(w|z')p(z')}. \quad (3)$$

To evaluate the conditional probabilities in the above equations, consider:

$$p(w|z) = \frac{p(z|w)p(w)}{p(z)}. \quad (4)$$

By marginalizing across u and d , we get:

$$p(w|z) = \frac{\sum_{u \in U} \sum_{d \in D} p(z|u, d, w)p(w|u, d)}{\sum_{u \in U} \sum_{d \in D} \sum_{w' \in W} p(z|u, d, w')}. \quad (5)$$

Finally, consider what $p(w|u, d)$ means. This is the probability of a given word occurring for a given email and person. Since the email and person are already specified the probability space is the one email. Therefore the probability is the number of times the word appears in the email divided by the number of words in the email. Therefore,

$$p(w|z) = \frac{\sum_{u \in U} \sum_{d \in D} p(z|u, d, w)n(d, w)}{\sum_{u \in U} \sum_{d \in D} \sum_{w' \in W} p(z|u, d, w')n(d, w)}, \quad (6)$$

where $n(d, w)$ is the number of times a word occurs in an email. Observe that since an email is the same regardless of which “author” is considered, it is sufficient to specify $n(d, w)$ so long as it is summed across all people. Furthermore, since the denominator sums across all words, the net effect is the quotient described previously. This equation extends

naturally to emails and users:

$$p(d|z) = \frac{\sum_{u \in U} \sum_{w \in D} p(z|u, d, w) n(d, w)}{\sum_{u \in U} \sum_{d' \in D} \sum_{w \in W} p(z|u, d', w) n(d, w)} \quad (7)$$

$$p(u|z) = \frac{\sum_{d \in D} \sum_{w \in W} p(z|u, d, w) n(d, w)}{\sum_{u' \in U} \sum_{d \in D} \sum_{w \in W} p(z|u', d, w) n(d, w)} \quad (8)$$

$$p(z) = \sum_{u \in U} \sum_{d \in D} \sum_{w \in W} p(z|u, d, w) \quad (9)$$

These equations form the expectation equation (Eqn. 3) and maximization equations (Eqns. 6–9) for Expectation-Maximization (EM). EM alternates two steps:

1. Assign random probabilities to $p(d|z)$, $p(w|z)$, $p(u|z)$, and $p(z)$ such that they produce probability distributions (i.e., the probabilities are all non-negative and sum to one).
2. Calculate all of the values for $p(z|u, d, w)$.
3. Using the values from Step 2, calculate the new values of $p(d|z)$, $p(w|z)$, $p(u|z)$, and $p(z)$.
4. Repeat Steps 2 and 3 until convergence.

5. Results

The Enron corpus was used as data in this research. During the investigation into the Enron scandal, the Federal Energy Regulatory Commission (FERC) made email from Enron publicly available. As a part of this process, it posted the email correspondence of 150 primarily senior employees on the World Wide Web. In addition to being valuable for prosecuting cases against Enron's senior management, this data has become a touchstone for research on email data mining techniques. Note that while it would be applicable to use Enron as a case study, this paper is a "proof of concept." As such, the Enron email corpus is used as data and only a small effort is made to uncover the principal actors involved in the Enron scandal. The entire Enron corpus was used; this consists of 245,904 emails made up of 54,147 stemmed words and 87,395 users, of which 34,885 were Enron employees. In addition, it was decided *a priori* that the corpus was made up of 48 categories based on previous research by McCallum and co-workers [7]. While the theoretical joint distribution of $p(z|d, w, u)$ could consist of approximately 5.5×10^{16} probabilities, the actual distribution for the corpus consisted of 3.4×10^9 probabilities.

Category 1	Category 14	Category 16	Category 27	Category 40	Category 45
Database 6.0%	Associate 1.6%	Outlook 5.2%	Image 0.4%	Migrate 2.3%	Ken 0.3%
Alias 4.0%	Analyst 1.6%	Migrate 4.5%	Expect 0.3%	Application 1.8%	Video 0.3%
Error 3.9%	Pilot 1.4%	Calendar 1.4%	Monitor 0.2%	Unify 1.1%	Lay 0.2%
Unknown 3.6%	Accept 0.4%	Mailbox 1.3%	Fool 0.2%	Directory 1.1%	Effort 0.2%
Variance 3.0%	Important 0.4%	Button 1.1%	Senate 0.2%	Enterprise 1.0%	Return 0.2%
Detect 2.4%	Opportunity 0.4%	Client 0.8%	Free 0.2%	Outage 0.9%	Corporation 0.2%
Log 2.2%	Feedback 0.3%	Journal 0.8%	Source 0.2%	Begin 0.8%	Board 0.2%
Parse 2.1%	Condition 0.3%	Web 0.7%	Security 0.2%	Log 0.7%	Rick 0.2%

Figure 3. PLSI-U sample categories (from the 48 available).

Nevertheless, a parallel algorithm had to be implemented to reduce the processing time per iteration to two hours. After running the algorithm, the data consistently converged to a mean square error (MSE) of less than 1×10^{-5} percent prior to 80 iterations. As a result, 80 was selected as a sufficient number of iterations.

Some of the words from the resulting categories are shown in Figure 3. These words have the highest conditional probabilities given the topic ($p(w|z)$). Although complete words are shown, they have been extrapolated from the word stems actually produced. Despite initial concerns that stemming might make some of the words difficult to determine (e.g., trying to determine the original word family that stemmed to 'thi'), the stemmed words that distinguished categories proved easy to identify. To produce a list that excluded common, non-distinguishing words, only words that appeared in at most five categories were used to define a category. While in general, this made the categories much easier to identify, the removal of some words, e.g., 'program' from Category 16, made some more difficult to understand. By manually examining the documents that had the highest conditional probabilities given the topic ($p(d|z)$), it became evident that, in general, the words with the highest probabilities did describe the categories well. For instance, in Category 1 many of the most likely documents concerned accessing scheduling databases. In Category 14, many promising documents described the Associate/Analyst Program, a mentoring program for new hires. Category 16 seemed to be about improving Enron's online presence through web traffic and email. And Category 40 describes a migration to new computer systems, backing up of directories, and a weekend outage.

Once the categories are resolved into words, the next step is constructing the social networks. For the implicit social network constructed from interests, this involves connecting every pair of people that had the same interest (Figure 4). For the explicit social network, consisting of the relevant emails, pairs of people are examined to see if they passed at least one email for the relevant topic between them; if so, a link is made be-

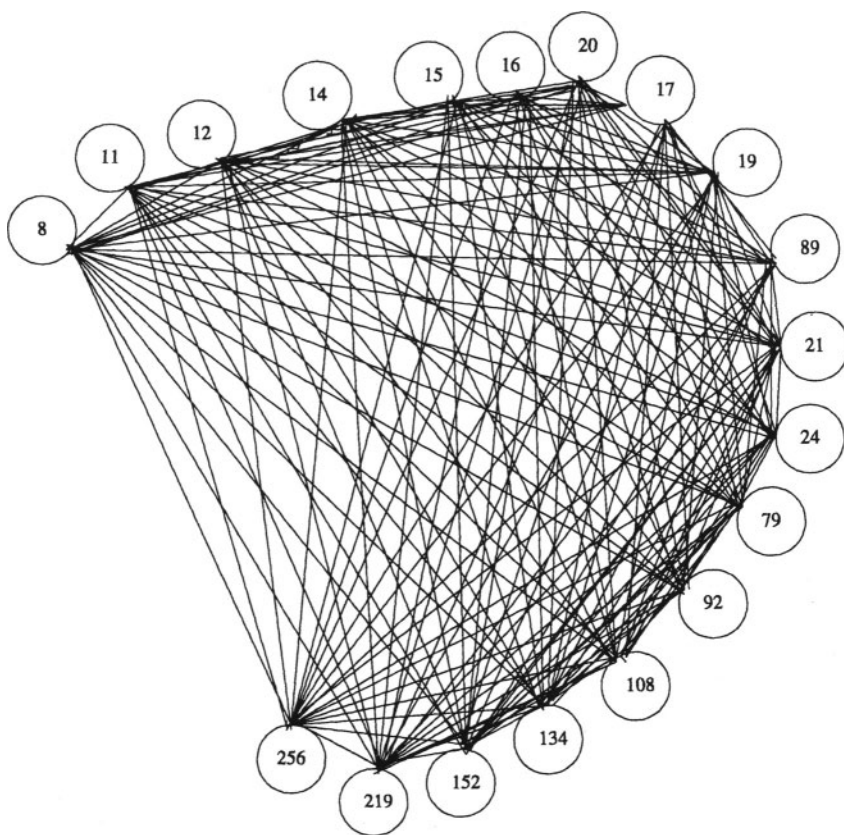


Figure 4. PLSI-U Enron implicit social network for Category 1.

tween them (Figure 5). In the case of Category 1, there is no one with an interest in Category 1 that does not have at least one email with at least one Enron employee who also has an interest in Category 1.

The final step is to consider the principal categories corresponding to each user. For instance, User 14920 was principally interested in Category 9 (62%) and secondarily interested in Category 38 (38%). Each Enron employee who sent emails was reviewed to see if there was at least one email sent or received for every category for which he/she had at least a 10% interest. The results were promising: across all 48 categories, only 103 employees showed up as having clandestine interests. This is even more remarkable when one considers that employees could show up as having clandestine interests in multiple categories (i.e., $34,885 \text{ employees} \times 48 \text{ categories} = 1,674,480$ possible clandestine interests). Even restricting the number to 34,885, means that less than 0.4% of the

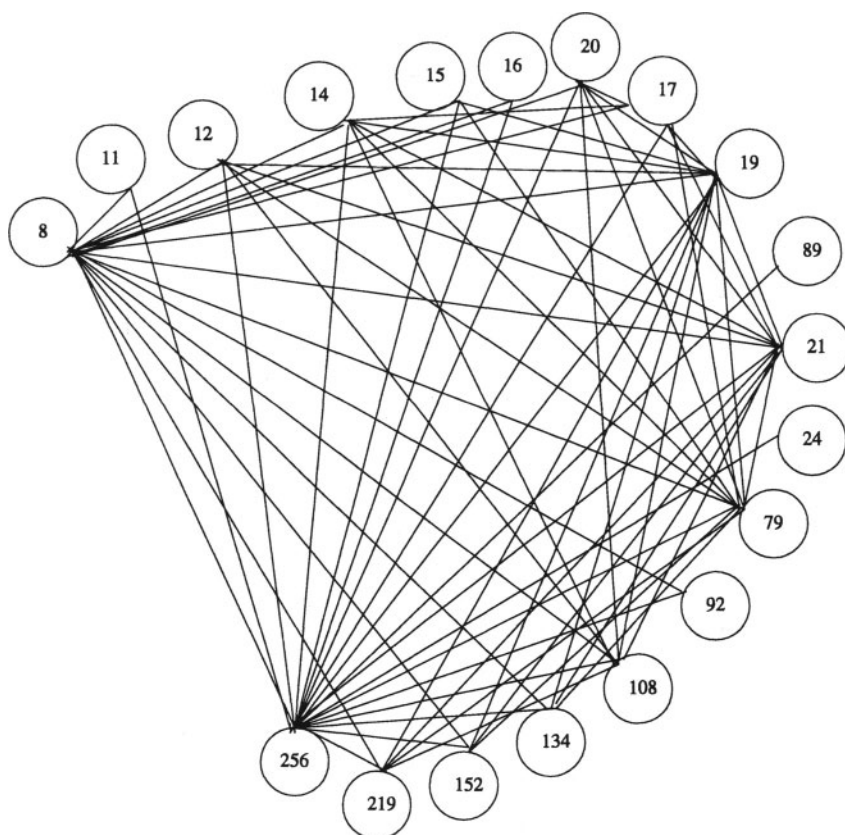


Figure 5. PLSI-U Enron explicit social network for Category 1.

people appeared to have clandestine interests. Of these, only 22 had a total of at least ten emails sent or received (the overall corpus average was 46). Therefore, it is very likely that the remaining 81 people are false positives having rarely or never used email. In the remaining 22 cases, the data does bear out that although PLSI-U shows a person with a strong interest in a specific topic, no email sent or received within Enron has that topic. For instance, one employee (User 14920) received 20 emails from a single non-Enron employee all pertaining to Category 9. Of the other four emails received and 15 emails sent within Enron, none had anything to do with Category 9 (Figure 6).

In addition to finding clandestine interests, the social networks generated are also useful. If investigators need to track down information on Category 1 (Figure 5), a good place to start would be User 256 since he is connected to everyone. If, on the other hand, they needed to start

User 14920	User 17594	User 46814	User 22446
15 Messages Sent 24 Messages Recv'd TOPIC 9 62% Topic 38 38%	15 Messages Sent 84 Messages Recv'd TOPIC 10 12% Topic 34 88%	2 Messages Sent 2102 Messages Recv'd Topic 2 42% TOPIC 12 10% Topic 30 48%	22 Messages Sent 57 Messages Recv'd TOPIC 23 18% Topic 41 41% Topic 43 2%
INTERNAL 15 Messages Sent 4 Messages Recv'd Principal Email Topics: 38 and 20	INTERNAL 15 Messages Sent 77 Messages Recv'd Principal Email Topics: 34, 9 and 11	INTERNAL 2 Messages Sent 335 Messages Recv'd Principal Email Topics: 2, 30 and 13	INTERNAL 22 Messages Sent 39 Messages Recv'd Principal Email Topics: 41, 43 and 15

Figure 6. PLSI-U sample users with clandestine interests (from the 22 extracted).

looking at possible suspects, perhaps Users 89 or 24 would be better as they have only a weak connection to other people interested in this topic. In this case, it might be suspicious that User 89, who has sent or received 1985 emails in total and has a 31% interest in this topic, has only emailed one other person about it.

6. Conclusions and Future Work

The results of the experiment show that the theory is sound. PLSI-U works well extracting topics from the email corpus and the simple mechanism for finding clandestine interests produces few false positives. While it would have been desirable to find Enron principals such as Kenneth Lay, Jeffrey Skilling and Andrew Fastow emerging as having clandestine interests, this did not occur. This may be because any questionable emails would have been sent to other people *inside* the organization, thus thwarting the algorithm described in this paper. While they did not emerge as having clandestine interests, it is informative that all three had only one or two topics of interest. All three individuals had a significant interest ($p(z|u) > 0.10$) in Category 45 (Figure 3) while Jeffrey Skilling and Kenneth Lay also had a significant interest in Category 27 (Figure 3). Therefore, while they do not appear to have clandestine interests, by examining the other people who also had a significant interest, additional people involved in the questionable business practices may emerge. Even Sherron Watkins, the Enron whistle blower, did not emerge as having a clandestine interest. By her own admission, she saw no gain from going outside the company [8]. Her only attempt at whistle blowing was to try to talk to Kenneth Lay herself.

Although our technique appears promising, much work remains to be done. While many categories were easy to identify based on the most probable words, some were not. A different model for extracting topics

might produce better results. Latent Dirichlet Allocation (LDA) has been shown to be a more general case of PLSI [2]. By not assuming that the mixture of topics in the corpus is the only possible mixture of topics, LDA has a better chance of describing previously unseen emails. Rosen-Zvi, *et al.* developed the Author-Topic Model [11] that expands on LDA by including clustering on individuals. This model may produce better results. Another promising technique is not to restrict the vocabulary to words found in a dictionary. Acronyms and words like “social” (for southern California) figure prominently in many Enron emails but they are excluded because they do not appear in a dictionary. By allowing these words to appear in the topics, the topics may become more identifiable.

The final cause for concern is the overloading of email. In this application, email is used to define the topics and to determine who is not revealing their interest in a topic. On the surface this should result in no clandestine interests since the only way someone is considered to have an interest in a topic is if they send or receive an email about it. The reason this is not the case is because, during the definition of topics, internal and external emails are considered, while during the search for clandestine interests only internal emails are used. By using a different data source for generating topics of interest, more clandestine interests may emerge. One logical data source is Internet activity. PLSI can easily morph from documents made up of words to web pages composed of hyperlinks [1]. While Internet activity logs were not available in the Enron case, they are generally available from the same sources that supply email logs.

References

- [1] D. Cohn and H. Chang, Learning to probabilistically identify authoritative documents, *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, California, pp. 167-174, 2000.
- [2] M. Girolami and A. Kaban, On an equivalence between PLSI and LDA (citeseer.ist.psu.edu/girolami03equivalence.html).
- [3] K. Herbig and M. Wiskoff, Espionage Against the United States by American Citizens 1947 – 2001, Technical Report, Defense Personnel Security Research Center, Monterey, California, 2002.
- [4] T. Hoffman, Probabilistic latent semantic indexing, *Proceedings of the Twenty-Second Annual ACM Conference on Research and Development in Information Retrieval*, 1999.

- [5] P. Keila and D. Skillicorn, Detecting Unusual and Deceptive Communication in Email, Technical Report, Queen's University, Kingston, Ontario, Canada, 2005.
- [6] S. Martin, A. Sewani, B. Nelson, K. Chen and A. Joseph, Analyzing behavioral features for email classification, *Proceedings of the Second Conference on Email and Anti-Spam*, 2005.
- [7] A. McCallum, A. Corrada-Emmanuel and X. Wang, Topic and role discovery in social networks, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2004.
- [8] B. McLean and P. Elkind, *The Smartest Guys in the Room*, Penguin, New York, 2003.
- [9] Merriam-Webster Collegiate Dictionary, Espionage (www.m-w.com/cgi-bin/dictionary).
- [10] RAND, Research and development initiatives focused on preventing, detecting and responding to insider misuse of critical defense information systems (www.rand.org/publications/CF/CF151/CF151.pdf).
- [11] M. Rosen-Zvi, T. Griffiths, M. Steyvers and P. Smyth, The Author-Topic Model for authors and documents, *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pp. 487-494, 2004.
- [12] S. Symonenko, E. Libby, O. Yilmazel, R. Del Zoppo, E. Brown and M. Downey, Semantic analysis for monitoring insider threats, *Proceedings of the Second Symposium on Intelligence and Security Informatics*, 2004.

Chapter 9

COLLUSION DETECTION USING MULTIMEDIA FINGERPRINTS

Anthony Persaud and Yong Guan

Abstract The large-scale distribution of digital multimedia over the Internet has seen steep increases in the numbers of criminal cases involving the unauthorized sharing and duplication of copyrighted multimedia content. Consequently, it is important to design reliable investigative techniques to combat unauthorized duplication and propagation, and to provide protection in the form of theft deterrence. Several fingerprint embedding schemes have been developed to combat single-user attacks. However, a new breed of attacks known as “collusion attacks” can defeat these schemes. Collusion attacks use the combination of multiple fingerprinted copies to create a new version of the multimedia artifact in which the underlying fingerprint is attenuated to render the colluders untraceable.

This paper proposes a wavelet-based fingerprinting scheme and a clustering algorithm for collusion attack detection and colluder identification. Experimental results show that the scheme can identify colluders while maintaining low miss rates and false accusation rates.

Keywords: Multimedia content, collusion attacks, multimedia fingerprinting

1. Introduction

Digital watermarks are often used to uniquely mark multimedia artifacts to help identify the original recipients of the artifacts. Watermarks are useful for investigating the unauthorized duplication and propagation of multimedia content. Also, they provide protection in the form of theft deterrence. Several fingerprint embedding schemes have been developed to combat single-user attacks, i.e., the duplication and dissemination of content by individuals. However, a new breed of attacks known as “collusion attacks” can defeat these schemes. Collusion attacks combine multiple fingerprinted copies of multimedia content, creating a new ver-

Please use the following format when citing this chapter:

Persaud, A., Guan, Y., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 105–118.

sion of the artifact in which the fingerprint is attenuated to render the colluders untraceable. In the highly interconnected digital world, collusion attacks have become a popular technique for defeating multimedia fingerprinting embedding schemes.

Most colluder detection techniques [14] rely on direct pattern correlation of the colluded fingerprint to the set of colluders. This assumes that the entire fingerprint can be recovered and that the entire set of possible colluders is known. However, such assumptions are not realistic in many real-world scenarios.

This paper proposes a wavelet-based multimedia fingerprinting technique and a clustering algorithm for collusion attack detection and colluder identification. The scheme engages wavelet transforms and statistical clustering techniques to detect and identify the colluders involved in a collusion attack. Experimental results show that it can identify colluders while maintaining low miss rates and false accusation rates.

Our approach has four main benefits. First, full fingerprint recovery is not required. Second, the colluder set is built from joint density observations, not from predictions. Third, the approach requires less computational overhead for colluder identification. Finally, the identification of colluder sets is independent of how multiple marked copies are combined in a collusion attack and the number of colluders involved.

The next section describes the framework used for multimedia forensics. Next, a threat model involving linear and non-linear collusion attacks is presented. The following sections describe the wavelet-based fingerprinting scheme and clustering algorithm. The final two sections present the experimental results and conclusions.

2. Multimedia Forensics Framework

The overall problem of multimedia fingerprinting and colluder identification has three components: (i) fingerprint embedding, (ii) fingerprint recovery and (iii) collusion attack detection and identification. Fingerprint embedding focuses on using robust methods to embed watermark information in different multimedia artifacts. Fingerprint recovery deals with the recovery of embedded watermarks. In some cases, only part of the watermark is recoverable due to various alteration attacks. Collusion attack detection and colluder identification involve analyzing correlations between the embedded watermark and the set of known watermarks that correspond to known users. Most watermarking schemes address fingerprint embedding and/or recovery. Our scheme is unique in that it addresses all three components of the overall problem.

Wavelets have proven to be the most effective and robust scheme for watermarking multimedia artifacts [8]. Fingerprint embedding typically uses wavelet transforms, e.g., Discrete Wavelet Transform (DWT), to decompose an image into sub-bands [15]. These sub-bands represent the image approximation coefficients, which can be combined with the watermark via additive embedding [13]. One of the main advantages of wavelet embedding is the ability to use higher energy watermarks in regions that are less sensitive to the human visual system. This provides a higher degree of robustness with little or no impact on quality [10].

Fingerprint recovery is similar to the fingerprint embedding process. DWT is used to decompose the artifact into its corresponding set of sub-band coefficients [15]. These coefficients are compared with the original non-watermarked coefficients to retrieve the differences in values [13]. The difference in values is the corresponding embedded watermark for each sub-band. The recovery process is performed for all sub-bands that may have an embedded watermark [10].

Collusion attack detection and colluder identification involve the application of watermark correlation methods. Correlations are computed between the recovered colluded fingerprint and the fingerprints of users who received the content.

To identify multimedia colluders, Chu, *et al.* [1] have proposed that the list of all possible colluder combinations be generated from the set of individuals who received the multimedia content. The fingerprint for each combination of possible colluders is compared to the retrieved watermark, and the colluders are identified by the process of elimination.

Judge and Ammar [5] have developed a hierarchical watermarking system called *WHIM*. The location of the potential colluders can be approximated using watermark verification through intermediary nodes.

Other detection techniques rely on direct pattern correlation of a colluded fingerprint with a combination of colluders [14]. Some of these techniques assume that the entire watermark is recoverable from the colluded copy.

3. Threat Model

Collusion attacks fall into two main categories: linear and non-linear attacks [14]. Interested readers are referred to [3, 16] for details about these attacks.

Collusion attacks typically synchronize multiple fingerprinted copies of a multimedia artifact and average the signal to produce a new copy. In some cases, colluders might use a variant of the average attack by adding a small amount of Gaussian noise ϵ to increase the attenuation

Table 1. Formulations of collusion attacks used in this study.

Attack	Formulation
Average	$\psi_x^{avg}(i, j) = \varepsilon + \sum_{n=1}^{ K } \psi_x^{(n)}(i, j) / K $
Minimum	$\psi_x^{min}(i, j) = \min(\{\psi_x^{(k)}(i, j)\}_{k \in K})$
Maximum	$\psi_x^{max}(i, j) = \max(\{\psi_x^{(k)}(i, j)\}_{k \in K})$
MinMax	$\psi_x^{minmax}(i, j) = \frac{1}{2} (\psi_x^{min}(i, j) + \psi_x^{max}(i, j))$
Randomized Negative	$\psi_x^{randneg}(i, j) = \begin{cases} \psi_x^{min}(i, j) & \text{with prob. } p \\ \psi_x^{max}(i, j) & \text{with prob. } 1 - p \end{cases}$

of the original fingerprint. Other attacks use statistical approaches to attenuate fingerprints. In most cases, the minimum, maximum and median values of each of the fingerprinted copies are analyzed to create a new less traceable copy. The collusion attacks considered in this study are shown in Table 1.

Let $|C|$ out of $|U|$ total users collude so that $C = \{c_1, c_2, \dots, c_n\}$, where $n = |C|$. Let $\psi'(i, j)$ represent the component of a colluded copy Ψ' at location (i, j) . Using $|C|$ copies, the component of $\psi'(i, j)$ is generated by combining the components of all $c \in C$ using any of the attacks formulated in Table 1.

$$c_1 = \begin{bmatrix} 2 & 1 \\ 3 & 8 \end{bmatrix}, c_2 = \begin{bmatrix} 4 & 2 \\ 5 & 4 \end{bmatrix} \quad (1)$$

$$\Psi^{avg} = \begin{bmatrix} 3 & 1 \\ 4 & 6 \end{bmatrix}, \Psi^{min} = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \quad (2)$$

$$\Psi^{max} = \begin{bmatrix} 4 & 2 \\ 5 & 8 \end{bmatrix}, \Psi^{minmax} = \begin{bmatrix} 3 & 1 \\ 4 & 6 \end{bmatrix} \quad (3)$$

$$\Psi^{randneg} = \begin{bmatrix} \psi^{min}(1, 1) & \psi^{max}(1, 2) \\ \psi^{max}(2, 1) & \psi^{min}(2, 2) \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 5 & 8 \end{bmatrix} \quad (4)$$

An example involving two colluders is presented in Equation 1. The terms c_1 and c_2 denote image information in matrix form possessed by the two colluders. The two colluders combine their watermarked copies to create a colluded copy Ψ' . In the following, we discuss each of the collusion attacks presented in Table 1 and specified by Equations 2–4.

3.1 Average Attack

This attack averages the corresponding components of each colluder's copy to produce a new value. In the example, component $\psi^{avg}(1, 1)$ is

calculated as $\frac{c_1(1,1)+c_2(1,1)}{2} = 3$. The colluded copy Ψ^{avg} is obtained by performing this computation for all the components of c_1 and c_2 .

3.2 Minimum Attack

This attack takes the corresponding minimum components of the $|C|$ fingerprinted copies of the colluders. In the example, component $\psi^{min}(1,1)$ is calculated as $\min(c_1(1,1), c_2(1,1))$. The colluded copy Ψ^{min} is generated by performing this computation for all the components of c_1 and c_2 .

3.3 Maximum Attack

This attack takes the corresponding maximum components of the $|C|$ fingerprinted copies used in the attack. Component $\psi^{max}(1,1)$ is calculated as $\max(c_1(1,1), c_2(1,1))$. Performing this computation for all the components of c_1 and c_2 yields Ψ^{min} .

3.4 MinMax Attack

In this attack, the averages of the minimum and maximum values of the corresponding components of the $|C|$ fingerprinted copies are used to produce the colluded copy. Component $\psi^{minmax}(1,1)$ is computed as the average $\frac{\psi^{min}(1,1)+\psi^{max}(1,1)}{2}$. Performing this computation for all the components of c_1 and c_2 yields Ψ^{minmax} .

3.5 Randomized Negative Attack

In this attack, the values of each of the components in the colluded copy take either the minimum or maximum values of the corresponding components of the $|C|$ fingerprinted copies. The value of a component of the colluded copy, $\psi^{randneg}(1,1)$, is set to the minimum value $\psi^{min}(1,1)$ with probability p and is set to $\psi^{max}(1,1)$ with probability $(1-p)$. Assume that $p = 0.5$, and suppose that ψ^{min} is chosen for $\psi(1,1)$ and $\psi(2,2)$, and ψ^{max} for the other components. The resulting $\Psi^{randneg}$, which is shown in Equation 4, is just one of the sixteen possible colluded results.

4. Fingerprinting and Colluder Identification

This section describes our collusion attack detection and colluder identification scheme. Wavelet watermarking is used to maximize fingerprint recovery. Statistical clustering techniques are used to accurately identify

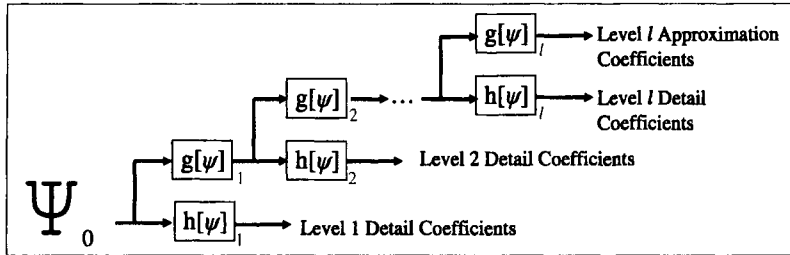


Figure 1. Level ℓ decomposition tree of a filter bank.

large colluder sets while minimizing $\max(|C' - C|, |C - C'|)$, where C is the set of real colluders and C' is the set of identified colluders.

4.1 Fingerprint Embedding

Multimedia artifacts can be represented as discrete signals, e.g., an image is represented as a matrix where each pixel location $\{i, j\}$ represents a color value. This property enables the use of the Discrete Wavelet Transform (DWT) [4]. DWT uses a decomposition process to embed fingerprint coefficients [2, 7]. This is done using band-pass arrays called “filter banks.” A filter bank is a series of high-pass and low-pass filters that partition the original signal into several components called “sub-bands.” The sub-bands can be recombined to recreate the original signal. The decomposition process can be applied to more than one level of decomposition because the wavelet transform is recursive in nature. At each level, the filter bank passes the input through a high-pass filter, $h[\psi]$, which provides the detail coefficients; and a low-pass filter, $g[\psi]$, which provides the approximation coefficients.

Figure 1 illustrates an ℓ -level decomposition tree using a filter bank. Note that the filters decompose the input into low and high frequencies at every level. Figure 2 shows a four-level recursive decomposition of the well-known *Lena* image.

Several robust wavelet-based watermarking methods have been developed [10]. This work employs the constant energy embedding technique because it requires the least amount of computation. In fact, the constant energy embedding technique is used as a baseline in most comparative studies [13]:

$$\psi'_\ell(i, j) = \psi_\ell(i, j) + \alpha \cdot f(i, j) \quad (5)$$

Embedding is performed by processing the multimedia artifact using the DWT with ℓ -levels of decomposition. After extracting the corresponding approximation coefficients, an additive embedding of the wa-

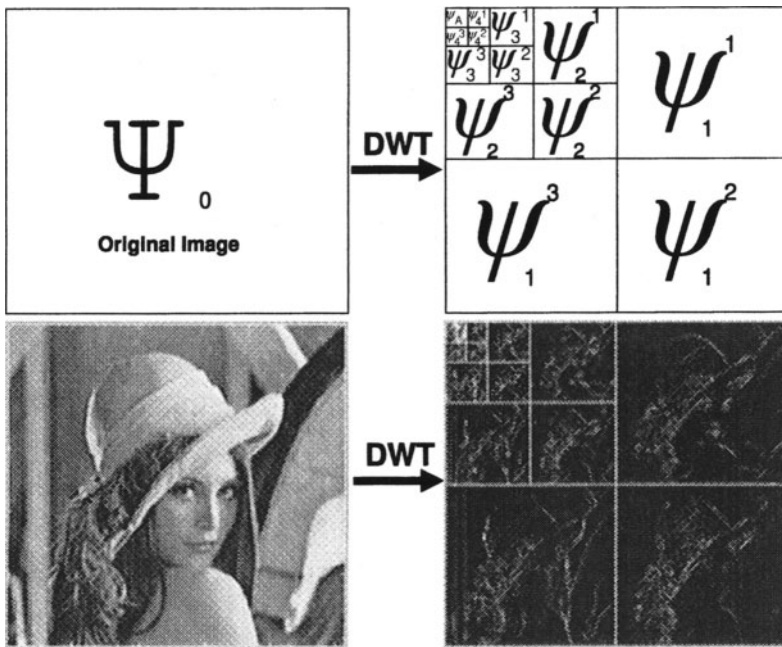


Figure 2. Four-level image decomposition (every ψ_i^n is a sub-band).

termmark is performed using the constant energy embedding technique in Equation 5. After the fingerprint has been embedded, the inverse DWT is performed to recreate the original artifact with the embedded watermark.

Let $\psi_\ell(i, j)$ be the component of the original image Ψ at location $\{i, j\}$. Let α be a global energy parameter that determines the fingerprint strength. Let f be the pre-computed fingerprint sequence, and ℓ specify the decomposition level of the coefficients used to embed the fingerprint. Spread-spectrum sequences [6] or orthogonal codes [12] can be used to generate a fingerprint f . In this study, a set of zero-mean Gaussian distributed random values is used to generate f . Our experimental results indicate that $\alpha = 0.1$ and $\ell = 4$ provide adequate fingerprinting strength with acceptable distortion.

$$\Psi_0 = \begin{bmatrix} 12 & 23 \\ 34 & 45 \end{bmatrix}, f = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \alpha = 2 \quad (6)$$

$$\Psi' = \begin{bmatrix} 12 & 23 \\ 34 & 45 \end{bmatrix} + 2 \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 14 & 23 \\ 36 & 47 \end{bmatrix} \quad (7)$$

Equations 6 and 7 provide an example of the computations involved in fingerprint embedding. Algorithm 1 formalizes the fingerprint embedding process.

Algorithm 1 : Fingerprint Embedding

```

1:  $\psi \leftarrow DWT(\Psi, \ell)$ 
2: if  $size(\psi) \leq size(f)$  then
3:   "error: fingerprint too large"
4: end if
5: for  $i \leftarrow 1$  to  $rows[\psi]$  do
6:   for  $j \leftarrow 1$  to  $columns[\psi]$  do
7:      $\psi'_\ell(i, j) = \psi_\ell(i, j) + \alpha \cdot f(i, j)$ 
8:   end for
9: end for
10:  $\Psi' \leftarrow iDWT(\psi', \ell)$ 
11: return  $\Psi'$ 

```

4.2 Fingerprint Extraction

The non-blind fingerprint extraction process is similar to the fingerprint embedding process. First, the original and fingerprinted artifacts are processed using DWT to extract the approximation coefficients. Next, the difference between these coefficients is calculated using Equation 8. The recovered fingerprint is denoted by f' .

$$f'(i, j) = \frac{1}{\alpha} \cdot (\psi'_\ell(i, j) - \psi_\ell(i, j)) \quad (8)$$

The fingerprint extraction process is formalized in Algorithm 2.

Algorithm 2 : Fingerprint Extraction

```

1:  $\psi \leftarrow DWT(\Psi, \ell)$ 
2:  $\psi' \leftarrow DWT(\Psi', \ell)$ 
3: for  $i \leftarrow 1$  to  $rows[\psi']$  do
4:   for  $j \leftarrow 1$  to  $columns[\psi']$  do
5:      $f'(i, j) = \frac{1}{\alpha} \cdot (\psi'_\ell(i, j) - \psi_\ell(i, j))$ 
6:   end for
7: end for
8: return  $f'$ 

```

4.3 Colluder Identification

After recovering the colluded fingerprint, f' , the correlation coefficient is calculated between two fingerprints, where f is the corresponding fingerprint of a user from a known database. Let $R(f)$ be the correlation

value between f' and f . The set R contains the correlation values between the colluded fingerprint and the fingerprint of each user.

Having determined the set of correlation values R and their corresponding users U , a statistical clustering technique can be applied to identify the colluders involved in the attack. Our scheme uses an iterative *2-means* clustering algorithm to obtain possible partitions in the set of colluders. This algorithm is a specialization of the well-known *k-means* algorithm [9]. The algorithm classifies the correlation values into two clusters, one is the set of detected colluders, and the other is the set of innocent individuals. Since higher correlation values indicate stronger relationships with the colluded fingerprint, the cluster with the highest mean value is considered to be the colluder set C' .

The clusters are partitioned by minimizing the Euclidean distance between every correlation value $R(f)$. The mean of a cluster is called its "centroid." In our variant of the algorithm, initial centroids are not selected randomly as in other algorithms, but are calculated based on the mean and standard deviation of the set R .

The algorithm computes two centroids for the entire data set R . During each iteration, a user f is assigned a group, C' or B , based on the shortest distance between $R(f)$ and one of the centroids. After all the users have been assigned to a group, the locations of the centroids are recalculated based on the members of each group. The process is repeated until the locations of the centroids do not change. The final result is the set C' , which contains the set of possible colluders involved in a collusion attack that yields f' .

The *2-means* algorithm is summarized below (Algorithm 3).

Algorithm 3 : 2-means Clustering

```

1:  $\bar{c} \leftarrow \text{mean}(R) + \text{stddev}(R)$ 
2:  $\bar{b} \leftarrow \text{mean}(R) - \text{stddev}(R)$ 
3: repeat
4:    $C' \leftarrow B \leftarrow \emptyset$ 
5:   for  $f \leftarrow 1$  to  $|R|$  do
6:     if  $|R(f) - \bar{c}| = \min(|R(f) - \bar{c}|, |R(f) - \bar{b}|)$  then
7:       Assign  $R(f)$  to set  $C'$ 
8:     else
9:       Assign  $R(f)$  to set  $B$ 
10:    end if
11:  end for
12:   $\bar{c}_{last} \leftarrow \bar{c}$  ;  $\bar{c} \leftarrow \text{mean}(C')$ 
13:   $\bar{b}_{last} \leftarrow \bar{b}$  ;  $\bar{b} \leftarrow \text{mean}(B)$ 
14: until  $\min(|\bar{c} - \bar{c}_{last}|, |\bar{b} - \bar{b}_{last}|) = 0$ 
15: return  $C'$ 

```

Table 2. Three iterations of the 2-means algorithm.

Iteration 1	$R(1) = 0.25$	$R(2) = 0.40$	$R(3) = 0.65$	$R(4) = 0.85$
$\bar{b} = 0.20$	0.05	0.20	0.45	0.65
$\bar{c} = 0.35$	0.10	0.05	0.30	0.50
Group Assignment	B	C'	C'	C'
Iteration 2				
$\bar{b} = 0.25$	0	0.15	0.40	0.60
$\bar{c} = 0.63$	0.38	0.23	0.02	0.22
Group Assignment	B	B	C'	C'
Iteration 3				
$\bar{b} = 0.325$	0.075	0.075	0.325	0.525
$\bar{c} = 0.75$	0.5	0.35	0.1	0.1
Group Assignment	B	B	C'	C'

We present a simple example to illustrate the clustering algorithm. In the example, the set R contains users $\{1, 2, 3, 4\}$, and $R(i)$ is the correlation value of user i . The initial values of the centroids, $\bar{b}$ and $\bar{c}$, are 0.20 and 0.35, respectively. During the first iteration of the algorithm, every point in R is assigned to a group based on the least distance to the corresponding centroid. Table 2 shows the group assignments after the first iteration.

After the initial group assignments, the values of the centroids are recalculated as the means of the members of each of the two groups. Therefore, $\bar{b} = 0.25$ and $\bar{c} = 0.63$. The process is repeated for the second iteration using the new centroids, and the new group assignments are presented in Table 2. At the end of Iteration 2, note that $R(2)$ has moved from C' to B . Again, the new centroids are calculated and the process is repeated.

The locations of the centroids are unchanged at the end of Iteration 3. Therefore, the algorithm terminates and C' contains the potential set of colluders because $\bar{c} > \bar{b}$.

This scheme is successful at determining the colluder set C' because colluders cannot obtain the value of the embedded fingerprint in their multimedia artifact. Therefore, they cannot determine which users from set C satisfy the condition $\text{corr}(f', f_i) = 1$ for an innocent user u_i . This becomes more difficult for colluders when orthogonal codes [12] and spread-spectrum watermarking [6] are used.

The algorithm is very practical because it treats the correlation values in R as random variables, and finds potential relationships based on joint densities. Therefore, the colluder set is built from observations not predictions. Furthermore, less computational overhead is involved because all possible colluder combinations do not have to be tested.

5. Experimental Results

Our collusion attack detection and colluder identification scheme was evaluated for collusion attacks on the *Lena* image (Figure 2). The fingerprints used were a sequence of pseudo-randomly generated Gaussian distributed values. MatLab 7.0 software was used for the computations.

A set of 400 fingerprinted copies of the *Lena* image were created. The fingerprints were embedded using the constant energy embedding technique and the Daubechies-6 filter. Set C contained a maximum of 200 colluders and set D contained a maximum of 200 innocent individuals. Colluded copies were generated for attacks from Table 1 involving two or more colluders from C . Four levels of decomposition were used for embedding, and the value of α was set to 0.10.

The experimental results show that the colluder identification scheme is highly effective against minimum, maximum and minmax attacks. The scheme works well for the randomized negative attack, although it degrades a little as the number of colluders increases. The scheme does not work quite as well against the average attack when the colluded fingerprint is closer to the mean of the distribution used to generate the fingerprint. The further the colluded fingerprint is from the mean, the better the scheme performs at identifying colluders who have used average attacks.

Figure 3 shows the miss rate of colluders, i.e., the number of colluders that are not detected, as the number of colluders increase.

Figure 4 shows the performance of the scheme with respect to false accusation rates as the number of colluders increase.

In summary, the colluder identification scheme is effective against minimum, maximum and minmax attacks. Also, it provides satisfactory protection for innocent parties in the event of randomized negative attacks.

6. Conclusions

The wavelet-based watermarking scheme and the statistical clustering technique proposed in this paper are useful for detecting and identifying individuals involved in collusion attacks. The wavelet-based watermarking technique provides a high fingerprint recovery rate. The 2-

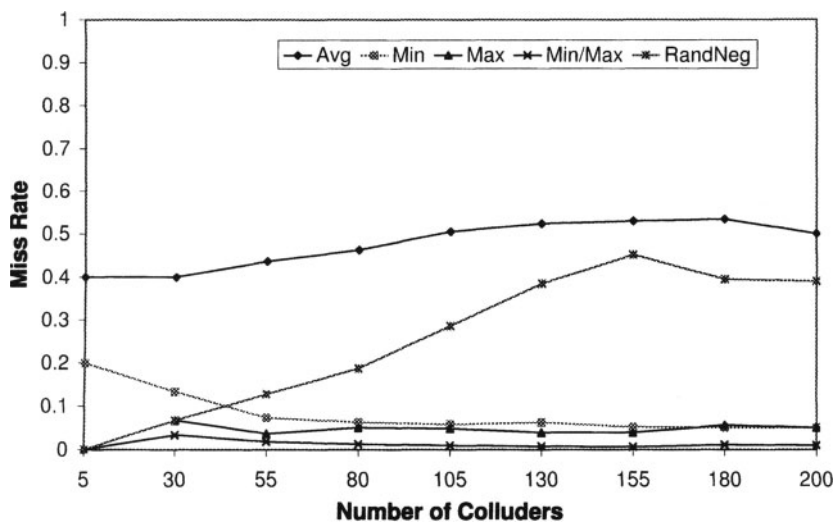


Figure 3. Miss rates with increasing numbers of colluders.

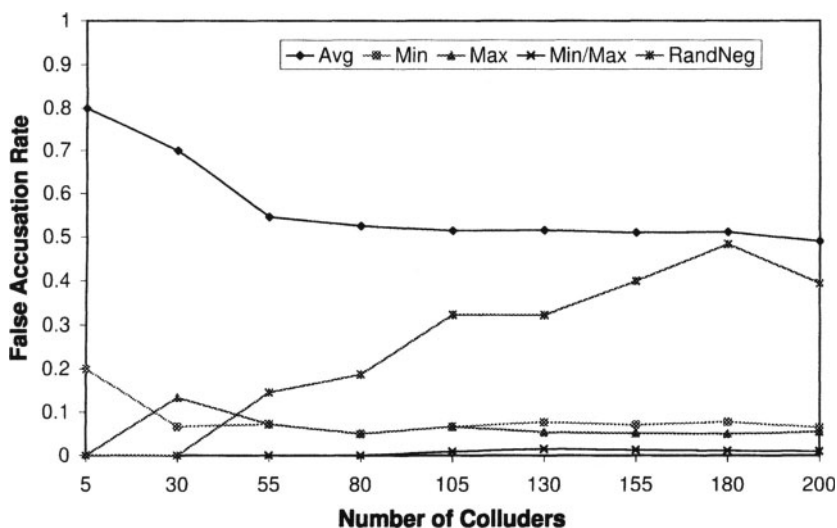


Figure 4. False accusation rates with increasing numbers of colluders.

means clustering algorithm is effective against collusion attacks because it builds the colluder set from value observations, not predictions. The experimental results show that the scheme is effective at thwarting common collusion attacks and determining colluder sets for a large number of colluders.

Acknowledgements

This research was partially supported by NSF Grant DUE-0313837, ARDA Contract NBCHC030107 and the GEM Fellowship Program. The authors also wish to thank Dr. Jennifer Davidson and anonymous reviewers for their advice and comments on earlier versions of this paper.

References

- [1] H. Chu, L. Qiao, K. Nahrstedt, H. Wang and R. Jain, A secure multicast protocol with copyright protection, *ACM Computer Communication Review*, vol. 32(2), pp. 42-60, 2002.
- [2] I. Cox, J. Bloom and M. Miller, *Digital Watermarking: Principles and Practice*, Morgan Kaufmann, San Mateo, California, 2001.
- [3] I. Cox, J. Kilian, T. Leighton and T. Shamoon, Secure spread spectrum watermarking for multimedia, *IEEE Transactions on Image Processing*, vol. 6(12), pp. 1673-1687, 1997.
- [4] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1992.
- [5] P. Judge and M. Ammar, WHIM: Watermarking multicast video with a hierarchy of intermediaries, *Computer Networks*, vol. 39(6), pp. 699-712, 2002.
- [6] J. Kilian, T. Leighton, L. Matheson, T. Shamoon, R. Tarjan and F. Zane, Resistance of Digital Watermarks to Collusive Attacks, Technical Report TR-585-98, Department of Computer Science, Princeton University, Princeton, New Jersey, 1998.
- [7] M. Kutter, F. Jordan and F. Bossen, Digital watermarking using multiresolution wavelet decomposition, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, pp. 2969-2972, 1998.
- [8] A. Lumini and D. Maio, A wavelet-based image watermarking scheme, *Proceedings of the International Symposium on Information Technonogy*, pp. 122-127, 2000.
- [9] J. MacQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281-297, 1967.
- [10] P. Meerwald and A. Uhl, A survey of wavelet-domain watermarking algorithms, *Proceedings of SPIE: Electronic Imaging, Security and Watermarking of Multimedia Contents III*, vol. 4314, 2001.

- [11] C. Shoemaker, Hidden bits: A survey of techniques for digital watermarking (www.vu.union.edu/~shoemakc/watermarking/watermarking.html), 2002.
- [12] Z. Wang, M. Wu, H. Zhao, W. Trappe and K. Liu, Anti-collusion forensics of multimedia fingerprinting using orthogonal modulation, *IEEE Transactions on Image Processing*, vol. 14, pp. 804-821, 2005.
- [13] C. Woo, J. Du and B. Pham, Performance factors analysis of a wavelet-based watermarking method, *Proceedings of the Third Australasian Information Security Workshop*, pp. 89-98, 2005.
- [14] M. Wu, W. Trappe, Z. Wang and K. Liu, Collusion resistant fingerprinting for multimedia, *IEEE Signal Processing Magazine*, pp. 15-27, March 2004.
- [15] X. Xia, C. Boncelet and G. Arce, Wavelet transform based watermark for digital images, *Optics Express*, vol. 3(12), pp. 497-511, 1998.
- [16] H. Zhao, M. Wu, Z. Wang and K. Liu, Nonlinear collusion attacks on independent fingerprints for multimedia, *Proceedings of the International Conference on Multimedia and Expo*, vol. 1, pp. 613-616, 2003.

Chapter 10

AUTHORSHIP ATTRIBUTION FOR ELECTRONIC DOCUMENTS

Patrick Juola

Abstract Forensic analysis of questioned electronic documents is difficult because the nature of the documents eliminates many kinds of informative differences. Recent work in authorship attribution demonstrates the practicality of analyzing documents based on authorial style, but the state of the art is confusing. Analyses are difficult to apply, little is known about error types and rates, and no best practices are available. This paper discusses efforts to address these issues, partly through the development of a systematic testbed for multilingual, multigenre authorship attribution accuracy, and partly through the development and concurrent analysis of a uniform and portable software tool that applies multiple methods to analyze electronic documents for authorship based on authorial style.

Keywords: Authorship attribution, stylometrics, text forensics

1. Introduction

The forensic importance of questioned documents is well-understood: did Aunt Martha really write the disputed version of her will? Document examiners can look at handwriting (or typewriting) and determine authorship with near miraculous sophistication from the dot of an “i” or the cross of a “t.” Electronic documents do not contain these clues. All flat-ASCII “A” characters are identical. How can one determine who made a defamatory, but anonymous, post on a blog, for example? Whether the authorship of a purely electronic document can be demonstrated to the demanding standards of a Daubert [25] hearing is an open, but important, research question.

Please use the following format when citing this chapter:

Juola, P., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 119–130.

2. Problem Statement

With the advent of modern computer technology, a substantial amount of “writing” today never involves pen, ink or paper. This paper is a good example—born as a PDF file, the first time these words see paper is in this bound volume. If my authorship of these words were challenged, I have no physical artifacts for specialists to examine.

Furthermore, the nature of electronic documents makes it substantially easier to “publish” or misappropriate them tracelessly or even to commit forgery with relative impunity. A network investigation at best only reveals the specific computer on which the document was written. It is almost impossible to figure out who was at the keyboard—who wrote it.

Chaski [6] describes three scenarios where it is both necessary to pierce the GUI and impossible to do so with traditional network investigations. In all three cases, there was no question about which computer the documents came from. Instead, the question was whether the purported authorship could be validated. The key question thus can be structured in terms of the message content. Can the authorship of an electronic document be inferred reliably from the message content?

3. Related Work

This section discusses research in authorship attribution, and the development of a test corpus for authorship attribution.

3.1 Authorship Attribution

Recent studies suggest that inferring the authorship of a document from its content is possible, but further research is necessary to meet the stringent Daubert criteria. The question of determining authorship by examining style has a long history. For example, Judges 12:5–6 describes the inference of tribal identity from the pronunciation of a specific word. Such *shibboleths* could involve specific lexical or phonological items; a person who writes of sitting on a “Chesterfield” is presumptively Canadian [7]. Wellman [27] describes how an idiosyncratic spelling of “touch” was used in court to validate a document.

At the same time, such tests cannot be relied upon. Idiosyncratic spelling or not, the word “touch” is rather rare (86 tokens in the million-word Brown corpus [20]), and it is unlikely to be found independently in two different samples. People are also not consistent in their language, and may (mis)spell words differently at different times; often the tests must be able to handle distributions instead of mere presence/absence

judgments. The discussion of methods to do this is an active research area: 70,400 hits turned up on May 4, 2006 on a Google search for “authorship attribution.” The increase from November 13, 2005 (49,500 hits) illustrates part of the continuing activity in this area in just six months.

Recent research suggests that statistical distributions of common patterns, such as the use of prepositions, may be universal enough to be relied upon, while still being informative. For this reason, scholars have focused on more sophisticated and reliable statistical tests. Specifically, Burrows [3–5] demonstrated that a statistical analysis of common words in large samples of text could group texts by author. Since then, many additional methods [1, 2, 6, 8, 10–12, 22–24] have been proposed. The current state of the art is an *ad hoc* mess of disparate methods with little cross comparison to determine which methods work and which do not. Or more accurately, because they all work at least reasonably well: under conditions discussed below, 90% accuracy is fairly typical for “good” methods. See [17] for details about which methods work the best.

Authorial analysis can even show more subtle aspects, such as the dates of documents. Figure 1 shows such an analysis [15] for a single author (Jack London), clearly dividing works written before 1912 from works that came later. The apparent division is a vertical line at about 3.14 on Dimension 1. Finding that a newly-discovered Jack London manuscript would be placed on the left-hand side of the diagram is strong evidence that it was written after 1912 as well.

3.2 Test Corpus Development

With the wide variety of techniques available, it is important but difficult to compare their power and accuracy. A fingerprint that can distinguish between Jack London and Rudyard Kipling, for example, may not work for Jane Austin and George Eliot. A proper comparison would involve standardized texts of clear provenance and known authorship on strictly controlled topics, so that the performance of each technique can be measured in a fair and accurate way. Forsyth [9] compiled the first benchmark collection of texts for validating authorship attribution techniques. Baayen [2] has developed a tighter series of texts produced under strictly controlled conditions.

To establish testing material, Baayen and co-workers at the University of Nijmegen elicited writing samples in Dutch from eight university students. The resulting 72 texts (8 subjects $\times$ 3 genres $\times$ 3 topics/genre) varied in length between 630 and 1,341 words (3,655–7,587 characters), averaging 907 words (5,235 characters) per text.

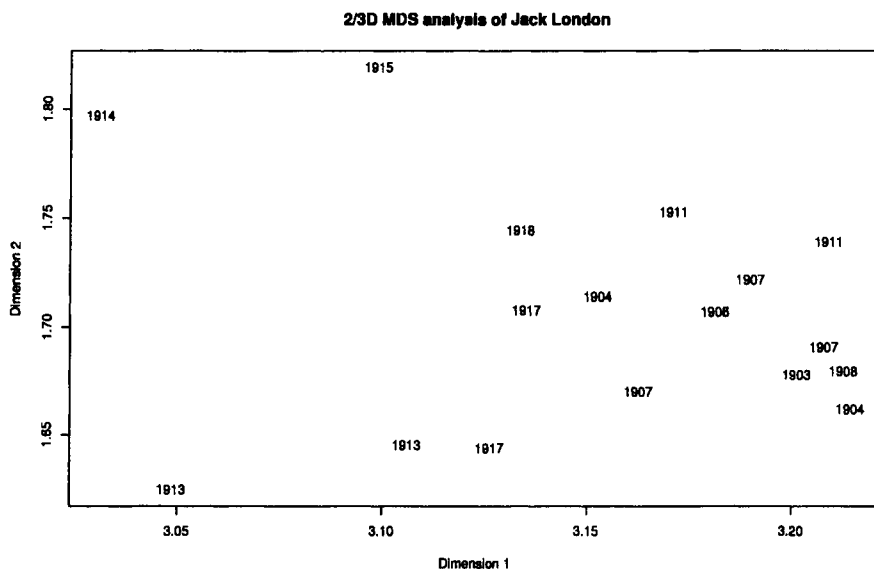


Figure 1. Spatial analysis of time development of Jack London's style.

This corpus has been comparatively analyzed using several different techniques. One of the most well-known authorship attribution techniques, proposed in [3] and later extended, is a principal components analysis (PCA) of the most common function words in a document. Another popular technique, linear discriminant analysis (LDA) [2], can distinguish among previously chosen classes, but as a supervised algorithm, it has so many degrees of freedom that the discriminants it infers may not be clinically significant. An alternative technique using measurements of cross-entropy has been independently proposed [12].

The question of which method is most accurate in this circumstance is easily answered: simply use all methods and compare the results. In particular, these methods have been tested [16] on the Baayen corpus. The software was presented with repeated trials consisting of triples containing all possible author pairs and disputed documents. Using this framework, function word PCA performed at essentially chance level, while function word LDA achieved 55% to 57% accuracy, depending upon the number of function words tabulated. Cross-entropy achieved up to 73% accuracy using a character-based model, and 87% accuracy across all pairwise comparisons using a word-based model.

From these results it can be concluded that under the circumstances of this test, cross-entropy and, in particular, word-based cross-entropy,

Table 1. Competition participants, affiliations and methods.

Name	Affiliation	Method
Baronchelli, <i>et al.</i>	Rome	Entropy-based informatic distance
Coburn	Middlebury	Contextual network graph
van Haltern	Nijmegen	“Linguistic Profiling”
Hoover	NYU	Cluster analysis of word frequencies
Hoover	NYU	Google search for distinctive phrases
Juola	Duquesne	Match length within a database
Lana and Amisano	UNIPMN	Common N-grams (two variants)
Kešelj and Cercone	Dalhousie	CNG with weighted voting
Kešelj and Cercone	Dalhousie	CNG-wv with reject
O’Brien and Vogel	Trinity/Dublin	Chi by degrees of freedom
Rudner	GMAC	Multinomial Bayesian Model/BETSY
Koppel and Schler	Bar-Ilan	SVM with linear kernel function
Stamatatos	Patras	Meta-classifiers via feature selection

is a more accurate technique for assessing authorship. However, the chance of a false assignment is an unacceptably high 13%.

4. Ad-hoc Authorship Attribution Competition

The authorship attribution studies raise an important follow-up question about the role of the test circumstances themselves. In particular, the test data was all in Dutch, the topics were very tightly controlled, and about 8,000 words of sample data per author were available. Would the results have been substantially different if the authors had written in English? If there had been 800,000 words per author, as might be the case in a copyright dispute involving a prolific author? Can the results of an analysis involving expository essays be generalized across genres, for example, to personal letters?

To answer these questions, the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities (ALLC/ACH) hosted an Ad-hoc Authorship Attribution Competition (AAAC) [13] (see Table 1). A standardized test corpus would not only allow researchers to test the ability of statistical methods to determine authorship, it would also allow “successful” methods to be distinguished from “very successful” methods. From a forensic standpoint, this would validate the science while establishing the standards of practice and creating information about error rates as Daubert requires.

Table 2. Detailed results (Problems A–G).

Team	A	B	C	D	E	F	G
baronchelli	3/13	3/13	8/9	3/4	1/4	9/10	2/4
coburn	5/13	2/13	8/9	3/4	4/4	9/10	1/4
halteren	9/13	3/13	9/9	3/4	3/4	9/10	2/4
hoover1	4/13	1/13	8/9	2/4	2/4	9/10	2/4
hoover2	4/13	2/13	9/9	4/4	4/4	10/10	2/4
juola	9/13	7/13	6/9	3/4	2/4	9/10	2/4
keselj1	11/13	7/13	8/9	3/4	2/4	9/10	3/4
keselj2	9/13	5/13	7/9	2/4	1/4	9/10	2/4
lana-amisano1	0/13	0/13	3/9	2/4	0/4	0/10	0/4
lana-amisano2	0/13	0/13	0/9	2/4	0/4	0/10	0/4
obrien	2/13	3/13	6/9	3/5	2/4	7/10	2/4
rudner	0/13	0/13	6/9	3/4	1/4	0/10	3/4
schler	7/13	4/13	9/9	4/4	4/4	10/10	2/4
stamatatos	9/13	2/13	8/9	2/4	2/4	9/10	2/4

4.1 Competition Setup

Competition materials included thirteen problems (see [13, 17] for details). These included a variety of lengths, styles, genres and languages, mostly gathered from the web but including some materials specifically gathered for the purpose. The participants (see Table 1) downloaded the anonymized materials and returned their attributions to be evaluated against the known correct answers.

4.2 Competition Results

The competition results (see Tables 2 and 3) were surprising at many levels. Some researchers initially refused to participate given the admittedly difficult tasks included among the corpora. Indeed, not all groups submitted results for all test problems. Problems for which no results were received were scored as 0/*N*.

For example, Problem F consisted of a set of letters extracted from the Paston letters. Aside from the very real issue of applying methods designed/tested for the most part for modern English on documents in Middle English, the size of these documents (very few letters, today or in centuries past, exceed 1,000 words) makes statistical inference difficult. Despite this apparent difficulty, almost all the groups were able to score 90% or better on this problem.

Similarly, Problem A was a realistic exercise in the analysis of student essays gathered in a first-year writing class—as is typical, no essay

Table 3. Detailed results (Problems H–M).

Team	H	I	J	K	L	M
baronchelli	3/3	2/4	1/2	2/4	4/4	5/24
coburn	2/3	2/4	1/2	2/4	3/4	19/24
halteren	2/3	3/4	1/2	2/4	2/4	21/24
hoover1	2/3	3/4	1/2	2/4	4/4	7/24
hoover2	3/3	4/4	2/2	2/4	4/4	7/24
juola	3/3	2/4	1/2	2/4	4/4	11/24
keselj1	1/3	3/4	1/2	2/4	4/4	17/24
keselj2	0/3	2/4	0/2	1/4	3/4	15/24
lana-amisano1	3/3	0/4	0/2	0/4	1/4	0/24
lana-amisano2	0/3	0/4	0/2	0/4	3/4	0/24
obrien	1/3	1/4	1/2	3/4	4/4	5/24
rudner	3/3	3/4	1/2	0/4	1/4	0/24
schler	2/3	3/4	2/2	1/4	4/4	4/24
stamatatos	1/3	3/4	1/2	2/4	3/4	14/24

exceeded 1200 words. From a standpoint of literary analysis, this may be regarded as an unreasonably short sample, but from a standpoint of a realistic test of forensic attribution and the difficult problem of testing the sensitivity of the techniques, these are legitimate.

Overall results from this competition were heartening. The highest scoring team (keselj1) had an average success rate of approximately 69%. In particular, Kešelj's methods achieved 85% accuracy on Problem A and 90% accuracy on Problem F, both acknowledged to be difficult and considered by many to be unsolvable. As a side note, Hoover identified a weakness in the problem structure. Since much of the data was taken from the web, a search engine such as Google could be used to identify many of the documents and, therefore, the authors. Hoover himself admits that this solution neither generalizes nor addresses the technical questions of stylometry.

All the participants scored significantly above chance on the problems for which they submitted solutions. Perhaps because most research focuses on English, performance on English problems tended to be better than those in other languages. More surprisingly, the availability of large documents was not as important to accuracy as the availability of a large number of smaller documents, possibly because they are more representative samples of an author's writing. Finally, methods based on simple lexical statistics performed substantially worse than methods based on N-grams or similar measures of syntax in conjunction with lexical statistics.

With regard to generalization and confidence issues, the findings are very good for the field as a whole. In general, algorithms that were successful under one set of conditions tended to be successful under other conditions. In particular, the average performance of a method on English samples (Problems A–H) correlated significantly ($r = 0.594$, $p < 0.05$) with that method's performance on non-English samples. Correlation between large-sample problems (problems with more than 50,000 words per sample) and small sample problems was still good, although no longer strictly significant ($r = 0.3141$). This suggests that the problem of authorship attribution is at least somewhat a language- and data-independent problem, and one for which we may be able to find wide-ranging technical solutions for the general case, instead of (e.g., in machine translation) having to tailor solutions with detailed knowledge of the problem/texts/languages at hand.

In particular, we offer the following challenge to researchers who are developing new forensic analysis methods: If you cannot get 90% correct on the Paston letters (Problem F), then your algorithm is not competitively accurate. Every well-performing algorithm studied in the competition had no difficulty achieving this standard. Statements from researchers that their methods do not work on small training samples should be regarded with some suspicion.

Unfortunately, another apparent result is that the high-performing algorithms appear to be mathematically and statistically (although not necessarily linguistically) sophisticated. The good methods have names that appear fearsome to the uninitiated: linear discriminant analysis [2, 26], orthographic cross-entropy [16], common byte N-grams [18], SVM with a linear kernel function [19]. Indeed, it may be difficult to explain to explain the underlying analysis techniques to a jury.

5. Future Developments

Because authorship attribution methods can be difficult to implement (and use) we cannot expect a casual user to apply these new methods without technical assistance. At the same time, the number of techniques proposed has exploded, which also limits the pool of available users.

This issue was addressed by Juola [14], who proposed a computational framework in which the different methods could be unified, cross-compared, cross-fertilized and evaluated to achieve a well-defined “best of breed.” During the past year, a proof of concept framework has been developed [17].

The framework postulates a three-phase division of the authorship attribution task, each of which can be independently performed. The three phases are :

- **Canonization:** No two physical realizations of events will ever be identical. Similar realizations are considered to be identical to restrict the event space to a finite set.
- **Event Set Determination:** The input stream is partitioned into individual non-overlapping events. At the same time, uninformative events are eliminated from the event stream.
- **Statistical Inference:** The remaining events can be subjected to a variety of inferential statistics, ranging from simple analysis of event distributions to complex pattern-based analysis. The results of this inference determine the results and confidence in the final report.

As an example of how this procedure works, we consider a method for identifying the language in which a document is written. We first canonize the document by identifying each letter (an italic *e*, a bold-face **e**, or a capital **E** should be treated identically) and producing a transcription. We then identify each letter as a separate event, eliminating all non-letter characters such as numbers or punctuation. Finally, by compiling an event histogram and comparing it with the well-known distribution of English letters, we can determine a probability that the document was written in English. A similar process would treat each word as a separate event (eliminating words not found in a standard lexicon) and comparing event histograms with a standardized set such as the Brown histogram [20]. The question of the comparative accuracy of these methods can be judged empirically. This framework allows researchers to focus on the important differences between methods and to mix and match techniques to achieve the best results.

The usefulness of this framework is verified by our prototype user-level authorship attribution tool. Currently, this tool coordinates and combines four different technical approaches to authorship attribution [4, 5, 12, 21]. The Java program combines a GUI atop the three-phase approach defined above. Users may select a set of sample documents (with labels for known authors) and a set of testing documents by unknown authors. Users are also able to select from a menu of event selection/preprocessing options and technical inference mechanisms. Three choices are currently supported: a vector of all the letters appearing in the sample/testing documents, a vector of all words so appearing, or a vector of only the fifty most common words/letters as previously selected, representing a restriction of the event model. Similarly, a variety of processing classes have been written to infer the similarity between

two different vectors. Authorship of the test document is assigned to the author of the most similar document.

As a specific example of application, we note that many of the AAAC methods relied on inferential statistics applied to N-grams. But N-grams of what? Juola's method was explicitly applied to N-grams of letters, van Halteren's to words or word "classes," Stammatatos' to "common words," and Koppel/Schler's to "unstable words." Therefore, we can, in theory, code Koppel's method for identifying unstable words as a separate instance of the event set class, then calculate inferential statistics using van Halteren's or Juola's method (as an instance of the inference class) possibly resulting in an improvement over any component method.

While this program is being refined, new methods are also being developed and improved. The AAAC data is still available on-line to permit people to test their methods, and we hope to incorporate new practices into our continuing study of best practices. At the same time, we will extend the functionality and user-friendliness of the system with the hope of making it more than a research prototype.

The AAAC corpus itself has some limitations that need to be addressed. For example, the mere fact that the data is on the web (in many cases, gathered from web-accessible public archives) gives an unfair advantage to any method that searches the web. Similarly, the multilingual coverage is unbalanced. The coverage of different genres is spotty and there are probably important issues that have not been addressed at all. We hope to create and offer a follow-up competition with an improved test corpus and more stringent analysis parameters.

6. Conclusions

Authorship attribution of electronic documents is an important problem in digital forensics. Recent developments in authorship attribution, including large-scale empirical experiments, are helping establish a set of best practices for analyzing questioned documents. Implicit in these experiments are an enhanced ability to create toolsets for analysis and the requirement to create new and more accurate experiments that validate the best practices.

References

- [1] S. Argamon and S. Levitan, Measuring the usefulness of function words for authorship attribution, *Proceedings of the Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities*, 2005.

- [2] R. Baayen, H. van Halteren, A. Neijt and F. Tweedie, An experiment in authorship attribution, *Proceedings of JADT 2002: Sixth International Conference on Textual Data Statistical Analysis*, pp. 29-37, 2002.
- [3] J. Burrows, Word-patterns and story-shapes: The statistical analysis of narrative style, *Literary and Linguistic Computing*, vol. 2, pp. 61-70, 1987.
- [4] J. Burrows, "an ocean where each kind. . . ." Statistical analysis and some major determinants of literary style, *Computers and the Humanities*, vol. 23(4-5), pp. 309-321, 1989.
- [5] J. Burrows, Questions of authorships: Attribution and beyond, *Computers and the Humanities*, vol. 37(1), pp. 5-32, 2003.
- [6] C. Chaski, Who's at the keyboard: Authorship attribution in digital evidence investigations, *International Journal of Digital Evidence*, vol. 4(1), 2005.
- [7] G. Easson, The linguistic implications of shibboleths, presented at the *Annual Meeting of the Canadian Linguistics Association*, 2002.
- [8] J. Farrington, *Analyzing for Authorship: A Guide to the Cusum Technique*, University of Wales Press, Cardiff, United Kingdom, 1996.
- [9] R. Forsyth, Towards a text benchmark suite, *Proceedings of the Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities*, 1997.
- [10] D. Holmes, Authorship attribution, *Computers and the Humanities*, vol. 28(2), pp. 87-106, 1994.
- [11] D. Hoover, Delta prime? *Literary and Linguistic Computing*, vol. 19(4), pp. 477-495, 2004.
- [12] P. Juola, The time course of language change, *Computers and the Humanities*, vol. 37(1), pp. 77-96, 2003.
- [13] P. Juola, Ad-hoc authorship attribution competition, *Proceedings of the Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities*, 2004.
- [14] P. Juola, On composership attribution, *Proceedings of the Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities*, 2004.
- [15] P. Juola, Becoming Jack London, to appear in *Journal of Quantitative Linguistics*.

- [16] P. Juola and H. Baayen, A controlled-corpus experiment in authorship attribution by cross-entropy, *Literary and Linguistic Computing*, vol. 20, pp. 59-67, 2005.
- [17] P. Juola, J. Sofko and P. Brennan, A prototype for authorship attribution studies, to appear in *Literary and Linguistic Computing*, 2006.
- [18] V. Kešelj and N. Cercone, CNG method with weighted voting, presented at the *Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities*, 2004.
- [19] M. Koppel and J. Schler, Ad-hoc authorship attribution competition approach outline, presented at the *Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities*, 2004.
- [20] H. Kučera and W. Francis, *Computational Analysis of Present-Day American English*, Brown University Press, Providence, Rhode Island, 1967.
- [21] O. Kukushkina, A. Polikarpov and D. Khmelev, Using literal and grammatical statistics for authorship attribution, *Problemy Peredachi Informatii*, vol. 37(2), pp. 96-198, 2000; translated in *Problems of Information Transmission*, MAIK Nauka/Interperiodica, Moscow, Russia, pp. 172-184, 2000.
- [22] T. Merriam, An application of authorship attribution by intertextual distance in English, *Corpus*, vol. 2, 2003.
- [23] J. Rudman, The state of authorship attribution studies: Some problems and solutions, *Computers and the Humanities*, vol. 31, pp. 351-365, 1998.
- [24] E. Stamatatos, N. Fakotakis and G. Kokkinakis, Automatic authorship attribution, *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 158-164, 1999.
- [25] Supreme Court of the United States, *Daubert v. Merrell Dow Pharmaceuticals*, 509 U.S. 579, no. 92-102, 1993.
- [26] 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.
- [27] F. Wellman, *The Art of Cross-Examination*, MacMillan, New York, 1936.

Chapter 11

LINKING INDIVIDUALS TO DIGITAL INFORMATION

Shelly Seier, David Greer and Gavin Manes

Abstract As computer crime increases in scope and magnitude, it is imperative to develop techniques that can link individuals to specific computers, computer programs and electronic documents. Unfortunately, scientific techniques that can establish these links are limited at best. This paper demonstrates that computer use characteristics can be employed to establish strong, legitimate links between individuals and digital information. Certain characteristics can be used to identify individuals. Other characteristics may be used to create profiles that assist in eliminating suspects and reducing the scope of investigations.

Keywords: Computer use characteristics, pattern analysis, identifying individuals

1. Introduction

On September 20, 2001 a distributed denial of service attack disabled vital navigation systems at the Port of Houston in Texas. The attack was traced to a computer in Aaron Caffrey's home in England, which contained an attack script with the words "coded by Aaron." Caffrey admitted to being a member of the "Allied Haxor Elite" group, and to hacking his friends' computers to "test their security" [1]. All the evidence pointed to Caffrey: the script contained his name, his machine executed the attack and he had the necessary technical expertise.

However, Caffrey claimed that his computer had been commandeered by another individual via a Trojan virus. Caffrey argued that this individual was responsible because the Trojan was in control of his computer at the time of the attack. Although a forensic examination of Caffrey's computer yielded no evidence of a Trojan, Caffrey was acquitted.

To prove that Caffrey was responsible for the crime, it was necessary to: (i) link Caffrey to the attack script, (ii) link Caffrey to his computer,

Please use the following format when citing this chapter:

Seier, S., Greer, D., Manes, G., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 131–140.

and (iii) link the execution of the attack script to Caffrey's computer. Linking the attack script to Caffrey's computer was accomplished by standard digital forensic techniques. However, linking Caffrey to the attack script and to his computer proved to be more difficult. This paper proposes that computer use characteristics can be employed to establish strong, legitimate links between individuals and digital information, which were missing in the Caffrey case.

2. Demonstrating Uniqueness in Individuals

Individuals possess unique characteristics such as fingerprints and DNA, which are often used as objective evidence [9]. When such evidence is not available, characteristics such as handwriting may be used to link an individual to a written document [6]. The uniqueness of an individual's handwriting comes from education, artistic ability, physiological development and preference. The slant, spacing and letter formation embody unique stylistic features that tend to become permanent over time [3, 4, 8]. Linguistics is an important component in written documents as spelling and word choice are distinct stylistic traits [2, 4, 5].

Handwriting samples fall into two categories: requested and non-requested. For requested samples, suspects agree to write a set of pre-determined words on paper. Non-requested samples are personal letters and notes created independent of the investigation.

It is important to obtain requested and non-requested writing samples to determine a match. Requested samples may be intentionally altered or may reflect a suspect's nervousness or excitement. Investigators can request lengthy samples to negate attempts at subterfuge; experienced handwriting analysts are usually able to discern intentional alterations of writing style. Typically, free-writing samples are better than requested samples as they manifest the true handwriting style.

Much like handwriting, the way an individual uses a computer is the result of education, artistic ability, physiological development and preference [7]. Because of the similarities between handwriting and computer use, handwriting sampling techniques could be used as a guide for developing computer use sampling techniques.

3. Digital Characteristics

If computer use can be monitored and measured, an investigator should be able to utilize a computer use sample in much the same way as a handwriting sample. A requested computer use sample would involve asking an individual to perform certain tasks on a computer with monitoring tools installed. A non-requested sample could be obtained

by monitoring an individual's computer use surreptitiously. There are several technical methods for monitoring and data collection; they can be categorized as active monitoring and passive monitoring.

Characteristics obtained using active monitoring include keystrokes, mouse use patterns and network use patterns. Other information can be obtained through passive methods such as traditional digital forensic investigations, undo history, application history, passwords and linguistic analysis of previously-typed documents [2, 5].

Active Monitoring

- **Keystrokes:** A keystroke logger can be used to record every keystroke made by an individual. These logs can determine the typing speed and style, including shortcuts, command line operations and program operations. For instance, one individual might use the backspace key to correct typing mistakes immediately, while another individual might leave the error for the spell checker to fix. A keystroke logger can also help determine accuracy by recording the use of the backspace and delete keys.
- **Mouse Distance/Patterns:** Each person has a unique way of using a mouse. Some may fidget while concentrating, others might trace words with the mouse as they read. Some may right-click for options, while others may use menus. Using the scroll bar instead of clicking and dragging to navigate inside a window is another distinctive mouse pattern.
- **Network Use Patterns:** Network monitoring tools can be used to log an individual's network use. Advanced monitoring tools can log every packet that is sent and received. This could help identify frequently visited Internet sites as well as the communications protocols that are used. Also, it could reveal the information content sent across the network.

Passive Monitoring

- **Undo History:** In relation to the frequency of error correction, the undo history may be unique for different individuals. One individual might use the undo backing order to retrieve an earlier paragraph, while another may just retype the entire paragraph. It may also be beneficial to know the set of circumstances under which an individual uses the undo function.
- **Dictionary/Word Choice:** Individuals may be identified based on the complexity of their word choice and use of a dictionary

or thesaurus. The frequencies of certain words and phrases in emails or chats may be a distinguishing characteristic, as well as expressions, shorthand and slang [2].

- **H4x0r T4lk:** Since the majority of computer crime is perpetrated by technologically-savvy individuals, it is important to understand computer-specific languages such as H4x0r T4lk (hacker talk), l33t (Leet) and chat shorthand. Certain groups of users substitute characters for letters, and often a character is substituted for the same letter every time. This consistency allows all the members of the group to understand the language. Knowing the styles used by hacker groups may indicate which groups to investigate in the event a crime has been committed.
- **Application History:** Individuals often favor one program or application over another. Some may use Internet Explorer, while others prefer Mozilla or Netscape. Many different chat clients, email clients and word processors exist, and most users have strong preferences among these programs. The combination of frequently used applications may be an identifying aspect of an individual.
- **Passwords:** Measuring password strength could eliminate certain individuals (e.g., those using weak passwords), thereby narrowing the list of suspects. Since many people use the same password for a variety of programs or functions, tracking password use over several computers might be beneficial in an investigation.

4. Evidence Collection Issues

Many of the same difficulties surrounding the collection of physical evidence hold for digital evidence, including the requirements of search warrants and application of forensically-sound evidence collection techniques. One of the more complex issues involves obtaining evidence from a suspect during an investigation. Active monitoring tools can provide valuable evidence, but these devices are invariably illegal without consent. Unfortunately, the time needed to obtain a warrant negatively affects the amount of monitoring that can be performed. Also, there is little chance that a suspect would agree to monitoring.

Collecting the requested evidence presents another set of difficulties. Typically, obtaining a requested computer use sample would involve asking an individual to perform certain tasks on a computer with the monitoring tools installed, but there is no guarantee the subject will not intentionally or inadvertently adjust his actions during the monitoring. As mentioned earlier, a possible solution is to collect large samples.

It is hard to determine the appropriate comparison of statistics with regard to requested and non-requested samples, as taking samples at different times might create bias. An individual's mood and energy level might affect the samples, as well as external forces such as network traffic loads, which could distort packet speed or chat room statistics. As in any statistical study, large samples and/or multiple samples, provide greater confidence in the results.

5. Experimental Results

To test whether digital characteristics are unique to each individual, an experiment was designed to monitor computer use. A laptop computer was loaded with keystroke monitoring software to capture each keystroke and the elapsed time between keystrokes. Then, a series of tasks was displayed on separate pages so the subject would not be able to prepare for the next task until the current task was completed.

The first task assigned to the subject was:

Open a Word document and begin typing a description of the weather today. Please try to type at least five lines of text.

If you cannot type five lines about the weather today, then write about how the weather has been since the beginning of this week.

Go to the next page.

The subject was asked to write about the weather to simulate a free thought process. As mentioned previously, requested handwriting samples do not reflect an individual's style as accurately as non-requested samples. This task allowed for some flexibility in word choice, grammar, sentence structure, punctuation, capitalization and spacing. Since there were no restrictions, the subject would be expected to type in a manner that accurately represents his typical style. From the monitor logs, an investigator would be able observe how the subject opened the word processor, how the subject formed sentences with respect to grammar, spacing, capitalization, punctuation, formatting and word choice, and how quickly the subject typed when he composed free-form sentences.

The second task assigned to the subject was:

Go to <http://www.nws.noaa.gov/>. Type "Tulsa" into the Local Forecast box on the top left and view the weather today.

Copy and paste the paragraph about today's weather into your Word document.

Go back to the Internet browser. Scroll to the bottom of the page and copy and paste the last paragraph about the weather a week from today.

Go to the next page.

This task was designed to monitor the use of the mouse and short-cut keys. The investigator would be able to observe how the individual opened the Internet browser, switched between windows, scrolled through windows, highlighted text, and copied and pasted text. The investigator could also note how the subject typed the web address into the address bar, and how he typed "Tulsa" in the text box.

The third task assigned to the subject was:

Retype the following paragraph into your document below the text you just pasted.

The preparedness guide explains thunderstorms and related hazards and suggests life-saving actions can take. With this information, you can recognize severe weather, develop a plan and be ready to act when threatening weather approaches. Contact your local National Weather Service Office for a variety of weather-related brochures.

Once the paragraph is completely retyped, save the document to the desktop as <yourname>.doc and close all windows.

You are now finished.

The purpose of this task was to demonstrate differences between typing freely-composed ideas and copying predetermined text. Both the typing speed and the attention to detail can be measured. For example, the subject might type "life-saving" with or without the hyphen. If the subject usually types two spaces after each sentence, he would probably not notice that all the sentences only have one space between them. Another detail to note is whether the subject tries to make logical sense of the sentences when retyping them. The first sentence reads "... and suggests life-saving actions can take" instead of "... and suggests life-saving actions you can take."

Another interesting result was the difference in typing speeds for free-form composition and predetermined text typing (Figure 1). Subjects tended to type at different speeds when they composed sentences as opposed to when they copied text. Although the differences in typing speeds varied, no individual had the same speed for both typing activities. It is possible to use this technique to distinguish individuals (and eliminate suspects) based on their typing speeds.

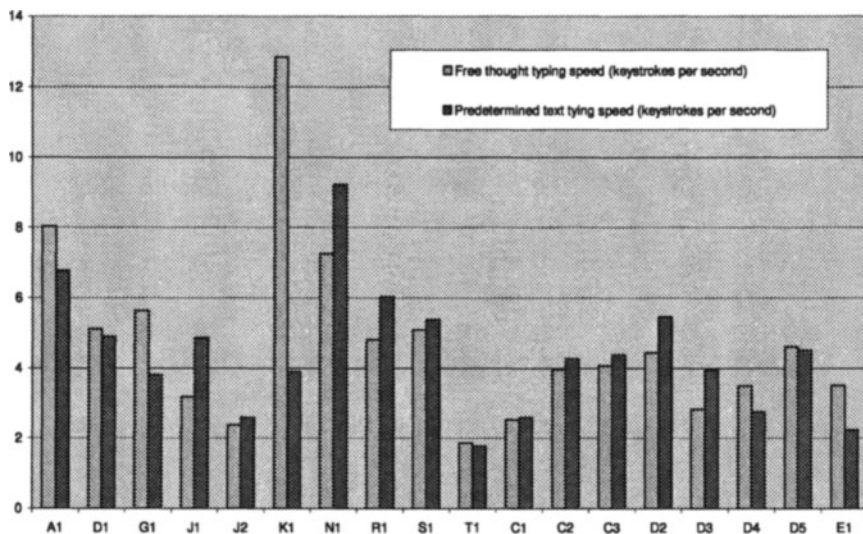


Figure 1. Keystrokes per second.

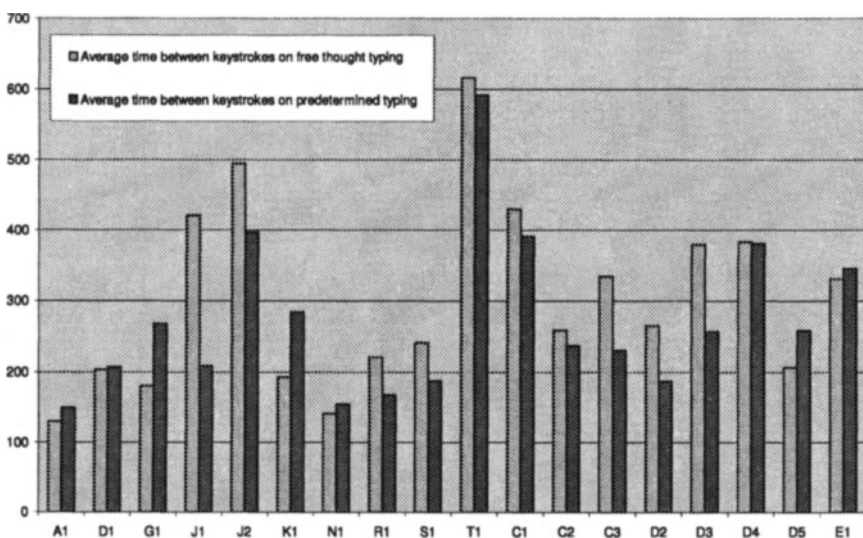


Figure 2. Time between keystrokes.

Figure 2 shows the differences in the time intervals between keystrokes for free-form composition and predetermined text typing. When comparing Figures 1 and 2, each subject differs considerably in terms of typing speed and keystroke intervals; thus, these could be useful identification characteristics.

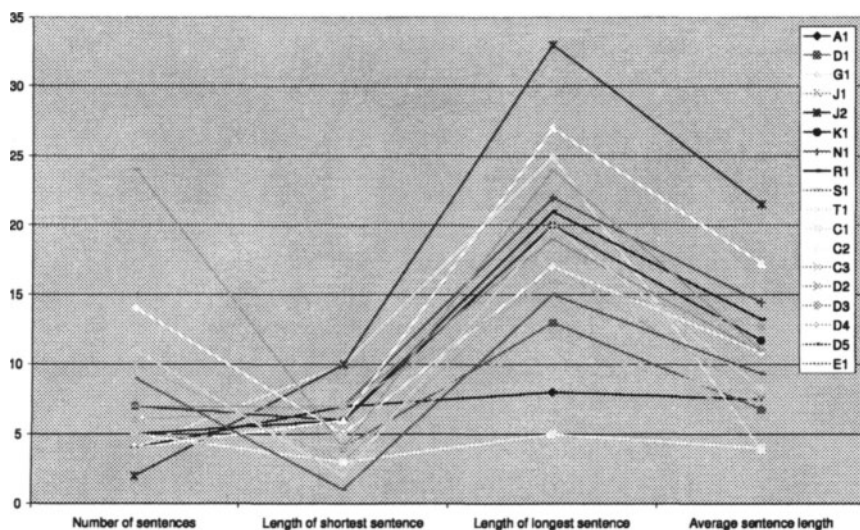


Figure 3. Sentence characteristics.

When given the same typing directions, each individual formatted the text differently. Figure 3 shows the differences in the numbers of sentences and sentence lengths. This particular task only required a small amount of text to be typed. It is expected that the differences would be much more significant for tasks that require a large amount of text to be typed.

Figure 4 indicates that each subject spent a different amount of time and corrected a different number of errors while performing the task. The lightest line represents the number of corrected mistakes compared to the elapsed time; the differences are unique for each subject. The difference between each subject's set of fastest-typed characters is also pronounced, and no two individuals had equivalent sets.

Groups of subjects showed similar results for characteristics such as name formatting when saving documents, use of the shift, control and backspace keys, use of shortcuts such as copy, paste and undo, window navigation (opening and closing), sentence and paragraph formatting, opening web pages, and typing in search criteria. Although these results may not distinguish a particular individual, they can be used to eliminate certain members of a group.

6. Future Work

The graphs presented in the previous section represent a modest subset of the analysis we have performed. Several other profiling points may

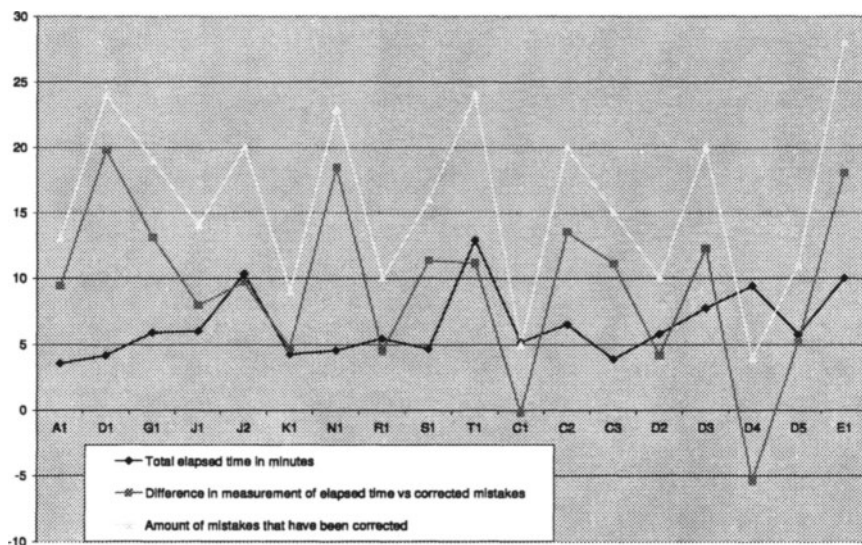


Figure 4. Differences in characteristics.

be developed from the test data. Analyzing the complete set of results from the tests is beyond the scope of this paper; however, additional results are available upon request.

Much like DNA evidence, some computer use characteristics may require the use of population profiling techniques [9]. Even if it is not possible to uniquely identify an individual, computer use characteristics can help identify a group of individuals as the target of an investigation, or exclude certain individuals from consideration.

Other techniques for identifying individuals include chat and network monitoring using keystroke monitors and surveillance tools. Unique characteristics involved in chat analysis include slang, acronyms, chatting style and typing speed, as well as the chat programs used, time of day and length of chat sessions. It might even be possible to identify individuals in chat rooms as they type their communications, and the monitoring could be performed remotely using chat room clients.

To monitor network use, packet analysis could be performed on various machines to track an individual's network use over time. Alternatively, two individuals could be given a specific question and asked to find the answer online. While they search for the answer, information pertaining to their browsing characteristics could be collected for on-line or off-line analysis.

7. Conclusions

The lack of robust scientific techniques for linking individuals to digital information hinders computer crime investigations and subsequent prosecution. Some of the principles underlying handwriting analysis can be applied to computer use characteristics to establish strong, legitimate links between individuals and specific computers, computer programs and electronic documents. The results can be used to identify individuals or to create profiles that may assist in eliminating suspects. Of course, this work is very preliminary; extensive research and statistical analysis are necessary before the techniques can be put to practice.

References

- [1] BBC News, Questions cloud cyber crime cases (news.bbc.co.uk/1/hi/technology/3202116.stm), October 17, 2003.
- [2] C. Chaski, Who's at the keyboard? Authorship attribution in digital evidence investigations, *International Journal of Digital Evidence*, vol. 4(1), 2005.
- [3] J. Olsson, *Forensic Linguistics: An Introduction to Language, Crime and the Law*, Continuum International Publishing Group, London, United Kingdom, 2004.
- [4] K. Ramsland, Document analysis (www.crimelibrary.com/forensics/literary), April 23, 2004.
- [5] V. Raskin, C. Hempelmann and K. Triezenberg, Semantic forensics: An application of ontological semantics to information assurance, *Proceedings of ACL 2004: Second Workshop on Text Meaning and Interpretation*, pp. 105-112, 2004.
- [6] G. Shpantzer and T. Ipsen, Law enforcement challenges in digital forensics, *Proceedings of the Sixth National Colloquium for Information Systems Security Education*, 2002.
- [7] E. Spafford and S. Weeber, Software forensics: Can we track code to its authors? *Computers and Security*, vol. 12(6), pp. 585-595, 1993.
- [8] S. Srihari, S. Cha, H. Arora and S. Lee, Individuality of handwriting, *Journal of Forensic Sciences*, vol. 44(4), pp. 856-872, 2002.
- [9] B. Weir, Population genetics in the forensic DNA debate, *Proceedings of the National Academy of Sciences*, vol. 89, pp. 11654-11659, 1992.

Chapter 12

USE-MISUSE CASE DRIVEN ANALYSIS OF POSITIVE TRAIN CONTROL

Mark Hartong, Rajni Goel and Duminda Wijesekera

Abstract Forensic analysis helps identify the causes of crimes and accidents. Determination of cause, however, requires detailed knowledge of a system's design and operational characteristics. This paper advocates that "use cases," which specify operational interactions and requirements, and "misuse cases," which specify potential misuse or abuse scenarios, can be used to analyze and link forensic evidence and create post-incident reconstructions. Use-misuse case analysis techniques involving non-probabilistic and probabilistic methods are described and applied to Positive Train Control (PTC) Systems – a network-based automated system that controls the movements of passenger and freight trains.

Keywords: Use-misuse case analysis, Bayesian belief networks, Positive Train Control (PTC) systems

1. Introduction

A forensic investigation involves the collection and analysis of evidence from the scene of an incident. Currently, investigators in the transportation sector, such as the National Transportation Safety Board (NTSB), make extensive use of the "Swiss Cheese Model" [23] to identify proximate and precursor causes of accidents. As an alternative, this paper presents a forensic analysis process rooted in the software development life cycle, which advocates that all the phases of system design should actively participate in and support incident investigation functionality.

The proposed forensic analysis process uses a software engineering technique called use-misuse case analysis, which examines system vulnerabilities and potential ways to exploit them [24–26]. Permissible interaction patterns provided by use cases constrain the scope of an investigation and convey knowledge about its operational domain in a succinct

Please use the following format when citing this chapter:

Hartong, M., Goel, R., Wijesekera, D., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 141–155.

manner, reducing the time spent by investigators to understand the domain and acquire evidence. Conversely, misuse cases, which incorporate known vulnerabilities and ways in which they can be exploited, provide investigators with alternative scenarios to pursue and identify potential evidence items.

Evidence found during a forensics examination may map completely (non probabilistically) to the evidence trait set defined by a misuse case. If the forensic evidence does not map completely, i.e., it is probabilistic in nature, techniques such as Bayesian Belief Networks (BBNs) [12, 14] can be used to obtain a probabilistic estimate about the misuse case that resulted in the incident.

In addition to describing the forensic analysis methodology, this paper compares its results with those from a traditional NTSB investigation of the June 2002 collision between Amtrak and MARC passenger trains in Baltimore, Maryland [18]. In fact, the NTSB recommendation relating to the use of Positive Train Control (PTC), a network-based system that conveys control messages for passenger and freight trains, is supported by the methodology.

The following section describes Positive Train Control (PTC) systems, use cases and misuse cases; it also shows how PTC functional requirements and potential misuse/abuse can be modeled via use-misuse cases. Section 3 discusses the NTSB forensic investigation of the Amtrak-MARC train accident, and shows how it can be viewed as an instance of a use-misuse case. Section 4 discusses the derivation of evidence traits from misuse cases. Section 5 describes a non-probabilistic mapping of evidence traits to misuse cases. Section 6 focuses on the probabilistic analysis of evidence using Bayesian Belief Networks (BBNs). The final section provides concluding remarks.

2. PTC System Use-Misuse Case Modeling

Positive Train Control (PTC) systems are increasingly used to ensure the safe operation of freight trains and passenger trains in the United States [8–11]. PTC offers significant enhancements in safety by providing for adequate train separation, enforcing speed restrictions, and protecting roadway workers. In a communication-based PTC implementation, functional subsystems are interconnected by a wireless network. Consequently, they are subject to the same vulnerabilities as other control systems that communicate using wireless networks. Although the vulnerabilities arise from common shortcomings of communicating subsystems, they manifest themselves in a specific control aspect by disrupting system functionality in a predictable manner.

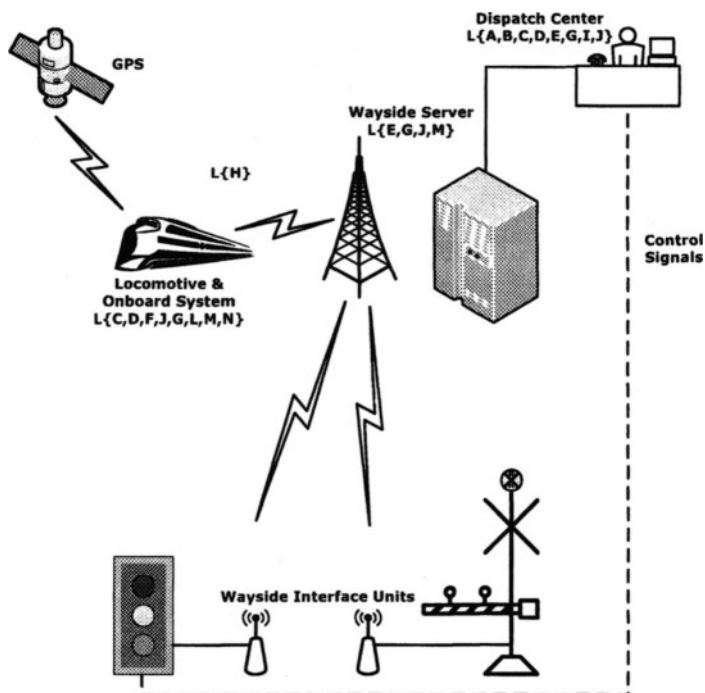


Figure 1. Simplified PTC architecture.

A simplified PTC architecture is presented in Figure 1. The architecture incorporates three major functional subsystems: wayside units, mobile units and a dispatch/control unit. The wayside units consist of elements such as highway grade crossing signals, switches and interlocks, and maintenance of way workers. The mobile units are locomotives and other rail equipment with onboard computers and location systems. The dispatch/control unit is the central office that runs the railroad. Each major functional subsystem consists of a collection of physical components implemented using databases, data communications systems and information processing equipment. Forensic evidence relevant to misuse cases for PTC systems is located in various subsystems. $L\{A\}$ through $L\{N\}$ in Figure 1 denote items of forensic evidence that relate to a specific misuse case. Note that these evidence items must be collected from multiple locations.

All PTC systems have the same core functional requirements. Table 1 specifies the functional requirements for various PTC levels [8, 9]. Note that each subsequent level imposes additional requirements.

Table 1. PTC levels and functionality.

Level	Functionality
0	None
1	Prevent train to train collisions; enforce speed restrictions; protect roadway workers and equipment
2	Level 1 functionality plus Digital transmission of authorities and train information
3	Level 2 functionality plus Monitor the status of all wayside switches, signals and protective devices in traffic control territory
4	Level 3 functionality plus Monitor the status of all mainline wayside switches, signals and protective devices, and additional devices (e.g., slide detectors, high water, hot bearings); implement advanced broken rail detection, roadway worker terminals for communications between dispatch and trains

In addition to functionality, PTC systems are also classified by the extent to which they augment railroad operations. Full PTC systems modify or replace the existing modes of railroad operation. Overlay PTC systems, on the other hand, provide their functionality while maintaining the existing modes of operation.

Deployed PTC systems operate with multiple components at the same time, forming a network of systems. Therefore, security and forensic aspects must be considered at the device level and at the network level. At the network level, it is necessary to identify sensitive network resources and components, and implement appropriate access control mechanisms. It is also important to prevent sabotage and misuse of PTC devices and network resources. The implementation of network management and security systems to protect, monitor and report on PTC systems without adversely impacting performance requires significant technical and financial resources.

2.1 Use Cases

Use cases capture how the users of a system will interact with the system. Ideally, they describe all possible interactions between an end user (person, machine or another system) and the system under consideration. Use cases also convey system requirements and constraints, and describe the essential features and rules under which the system and users operate. Use case diagrams are graphical instantiations of use cases (see Figure 2).

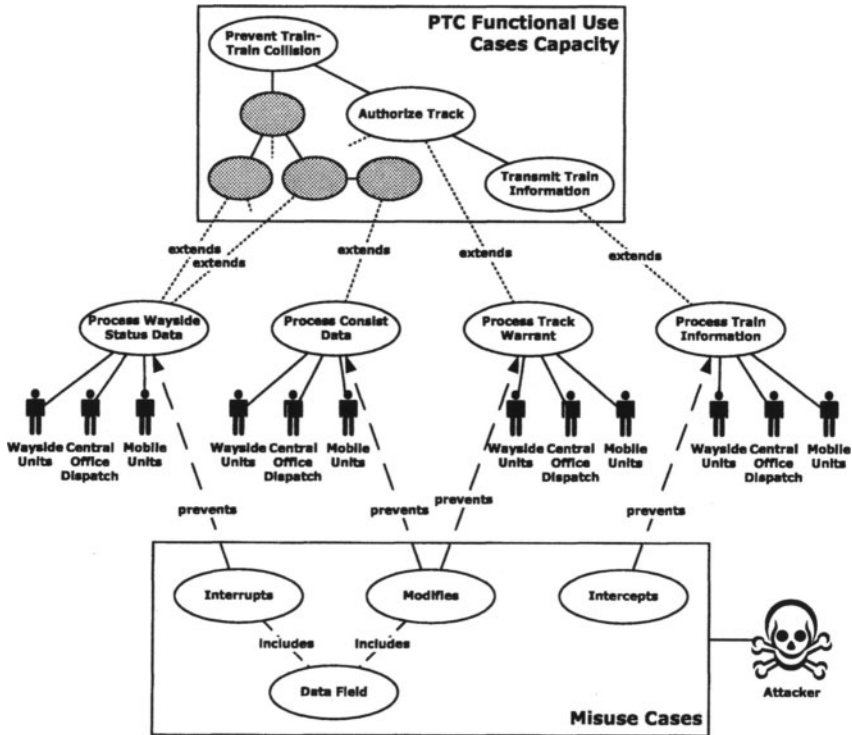


Figure 2. Use-misuse case relationships.

2.2 Misuse Cases

Misuse cases [24, 25] specify the external view of system behavior with respect to interactions between actors and/or mal-actors and the system. PTC actors in the use cases in Figure 2 include office/dispatch, wayside and mobile unit operators. Potential mal-actors are abstracted to a single attacker in the misuse cases. Figure 2 demonstrates how misuse cases can affect four use cases: (i) Process Wayside Status Data, (ii) Process Consist Data, (iii) Process Track Warrant, and (iv) Process Train Information. All actors – and the attacker — communicate by exchanging messages using the PTC system. Note that message formats used in PTC systems are implementation dependent.

A secure PTC system ensures that the safety services provided for the various PTC functions are available even in an exploitable communications environment. The repeated application of use-misuse case analysis to the functional requirement, Prevent Train-Train Collision, for exam-

ple, yields the security requirements of confidentiality, integrity, availability, authentication, accountability and identification. This process is repeated as required for each PTC functional requirement in Table 1. By analyzing additional postulated misuse cases, it is possible to obtain the aggregated security requirements for PTC Levels 1 through 4.

In the following, we describe a specific misuse case, Modify Track Warrant, in the format specified by Sindre and Opdahl [25]. In particular, we discuss how the misuse case leads to the generation of security requirements, and the establishment of a set of defining evidence traits needed for forensic analysis.

Misuse Case: Modify Track Warrant

Summary: Track warrant message is modified. This message conveys information that prevents train to train, train to on-track equipment, on-track equipment to on-track equipment, and train to roadway worker collisions.

Basic Path: The track warrant message is transmitted from the office/dispatch system to a mobile unit. The CRC is modified while the message is en route, rendering the message invalid. The mobile unit receives the invalid message. Acting on the invalid message, the mobile unit strikes another train, a track vehicle or roadway workers.

Alternate Paths: The track warrant message is relayed through the wayside subsystem and, during transmission, the CRC of the message is modified between the office/dispatch subsystem and the wayside subsystem, or the wayside subsystem and the mobile unit.

Capture Points: The track warrant message is invalid because one or more fields are modified: source, type, message payload and message identifier.

Triggers: Attacker places a transmitter within range of the subsystem's receiver and/or transmitter.

Attacker Profile: Attacker can capture the original message, read and interpret the message, modify one or more message fields, and retransmit the message.

Preconditions:

1. The office/dispatch subsystem is transmitting a track warrant message to a mobile unit.
2. The office/dispatch subsystem and the mobile unit subsystem are operating normally.

Post Conditions (Worst Case):

1. The mobile unit receives an invalid track warrant message, causing a train to train, train to on-track equipment, track to on-track equipment or train to roadway worker collision.
2. Unauthorized modifications of track warrant messages disable accountability and non-repudiation of specific operational restrictions and authorizations for potentially high hazard events such as commingling of roadway workers and trains.

3. An invalid track warrant message halts mobile units at the limits of its authority, producing a significant operational and safety impact.

Post Conditions (Best Case):

1. Message origin information is authenticated and data integrity is maintained.
2. Track warrant message modifications are identified and isolated.
3. Two entities do not commingle although they operate on altered track warrant messages.

Business Rules:

1. Only the office/dispatch subsystem originates valid track warrant messages.
2. The office/dispatch subsystem may push a valid track warrant message to a mobile or wayside subsystem.
3. The mobile subsystem may pull or request pulling a valid track warrant message from the wayside subsystem or the office/dispatch subsystem.
4. The wayside subsystem may pull a valid track warrant message from the office/dispatch subsystem only after the receipt of a request to pull a track warrant message from a mobile subsystem unit.

3. Railway Accident Investigation

Before discussing our methodology, we illustrate how collected evidence and pre-analyzed use-misuse cases can be used to determine probable cause in a documented railway accident investigation. We consider the June 2002 collision of Amtrak and MARC trains in Baltimore, Maryland [18].

We assume that pre-defined use-misuse cases associated with the operation of a locomotive by an engineer are already available. These use-misuse cases are created prior to an accident by analyzing the engineer's interactions with the locomotive, wayside systems and other systems, and identifying possible failure modes. Figure 3 presents a portion of the use-misuse case diagram for locomotive operation.

Forensic evidence gathered by investigators after an accident may include locomotive event recorder data, statements from the crew and other witnesses, recordings from the dispatch center, test data related to the operation of wayside devices (switches, signals, etc). The investigation of the Amtrak-MARC accident revealed that the engineer concentrated on monitoring speed to prevent flat spots, did not see the stop signal, and did not know how to apply the direct release air brakes.

Upon marking these facts in Figure 3 and tracing back to the root node for each misuse case, it is determined that the root causes of the accident were task fixation and lack of knowledge. Furthermore, failure to counter the misuse cases, task fixation and lack of knowledge,

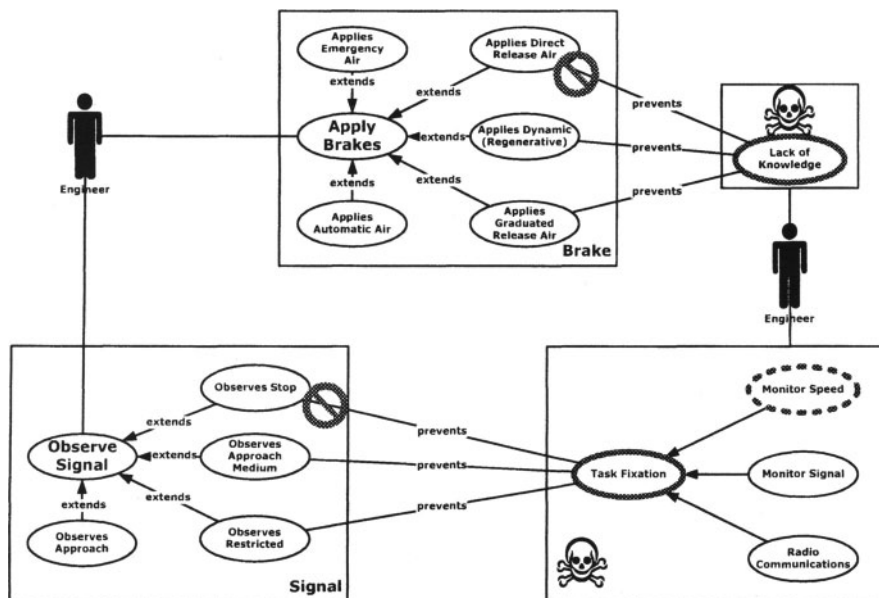


Figure 3. Amtrak-MARC use-misuse case diagram.

also contributed to the accident. These results closely match the NTSB probable cause findings of loss of situational awareness, lack of familiarity and proficiency, and absence of a PTC system [18].

4. Evidence Traits

Evidence traits provide detailed definitions of use cases and misuse cases. The notion of an evidence trait is a simple refinement of the concept presented in Section 3. However, unlike in Section 3, where the evidence gathered represents an entire use case or misuse case, the granularity of evidence is increased to include other attributes of use cases, such as pre conditions, post conditions and business rules. These evidence traits are captured from textually-specified use-misuse cases by analyzing nouns and verbs via a technique called “noun-verb extraction” [17]. Noun-verb extraction identifies specific characteristics of use cases and misuse cases that could represent evidence, i.e., behavior that is directly observed or conclusively inferred from observed behavior. The extraction process can be done manually by an engineer or by using specialized tools [20].

Table 2 presents the results of noun-verb extraction for the misuse case: Modify Track Warrant. The extractions identify the forensic ev-

Table 2. Evidence traits for PTC system.

Trait	Description
A	Text of message conveys authorization to occupy section of track
B	Message transmitted by office/dispatch system to mobile unit
C	CRC of message modified en-route, rendering message invalid
D	Mobile unit strikes another train, track vehicle or roadway workers
E	Message relayed to a wayside subsystem
F	Message invalid due to one or more modified fields: (i) source, (ii) type, (iii) payload, (iv) identifier
G	Attacker's transmitter placed within range of subsystem's receiver and/or transmitter
H	Attacker captures, reads, interprets, modifies and retransmits message
I	Office/dispatch subsystem transmits message to a mobile unit
J	Office/dispatch subsystem and mobile unit operates normally
K	Office/dispatch subsystem originates messages
L	Invalid message halts mobile units at limits of its current authority
M	Unauthorized modifications of messages disable accountability and non-repudiation of operational restrictions/authorizations for potentially high hazard events
N	Invalid message received causing train to train, train to track equipment, track to on-track equipment, train to roadway worker collisions

idence traits. Note that the physical locations of Evidence Traits A through N in Table 2 are identified in Figure 1.

5. Non-Probabilistic Forensic Analysis

Bogen and Dampier [1] and Pauli and Xu [21] have developed strategies for planning digital forensic examinations by systematically organizing, analyzing and identifying the most relevant concepts in a security incident, and determining the relations between these concepts. Our methodology, on the other hand, uses digital evidence to create an identifying signature. This signature is then matched with a set of previously identified misuse cases to identify a specific misuse case. Alternatively, the signature may be used to formulate a previously unidentified misuse case and generate the associated security requirements.

In non-probabilistic forensic analysis, the collected evidence represents a single identifying signature that has a one-to-one correspondence with a specific misuse case. This signature assists in mapping to a misuse case once evidence has been discovered. Also, it provides the forensic investigator with an initial set of traits and their locations, which facilitate the investigation.

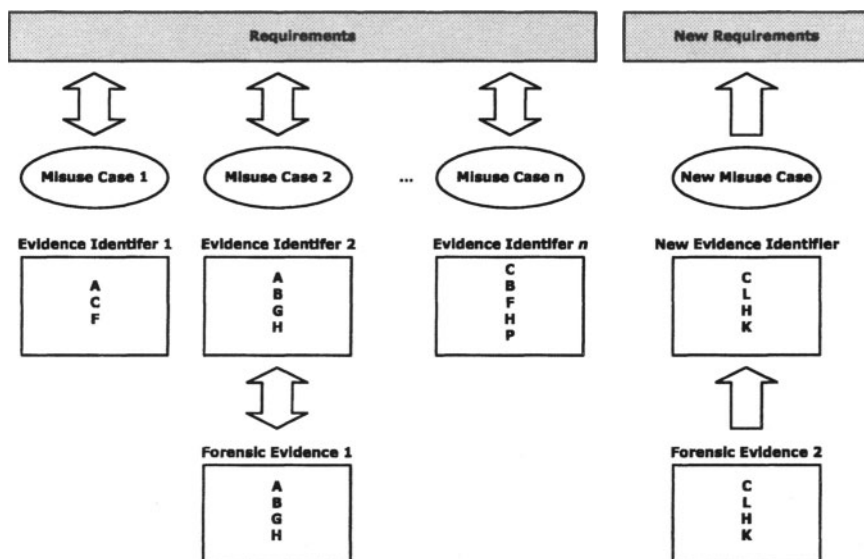


Figure 4. Non-probabilistic process.

The non-probabilistic process is outlined in Figure 4. The set of defined misuse cases (Misuse Case 1 to Misuse Case n) generates a corpus of known security requirements. Each misuse case is uniquely fingerprinted in terms of its own set of evidence traits (Evidence Identifier 1 to Evidence Identifier n). Forensic evidence obtained from an incident is compared to each fingerprint. In Figure 4, the collected evidence (Forensic Evidence 1) maps to the evidence trait Evidence Identifier 2. Evidence Identifier 2 uniquely identifies Misuse Case 2. This validates the requirements of Misuse Case 2 because the evidence is proof that Misuse Case 2 has occurred.

On the other hand, forensic evidence (Forensic Evidence 2) collected from another incident does not correspond to any existing evidence traits. Consequently, a new evidence trait set (New Forensic Identifier) is created for a new misuse case (New Misuse Case). The PTC system design may have to be adapted to account for the misuse case that yields the new set of evidence traits and identifies new security requirements. Note that the new misuse case and the associated requirements integrate into the corpus of known conditions.

6. Probabilistic Forensic Analysis

The evidence available to an investigator is often incomplete and may not match a misuse case fingerprint. A probabilistic match is required in such a situation. Bayesian Belief Networks (BBNs) [12, 14] offer a promising approach for probabilistically matching forensic evidence with evidence traits and associated misuse cases.

BBNs are directed acyclic graphs that capture probabilistic relationships between variables. Using a BBN to capture probabilistic relationships has several advantages. BBNs do not require exact or complete historical knowledge about the relationships between the variables. They may be created based on the available knowledge; however, as additional evidence is gathered, the relationships between variables may be adjusted to reflect the new evidence and compensate for missing information. Their easily understandable graphical structure simplifies their creation, modification and maintenance by domain experts, while providing opportunities for efficient computation. BBNs also support the determination of cause from effect just as easily as effect from cause; this enables them to be used to reason in a forward or backward manner with the available data.

Figure 5 shows an example BBN. An investigator is assumed to have discovered forensic evidence items A, C, F and G (represented by “true” conditions or complete (100%) certainty). On the other hand, the forensic evidence items B, H and L have not been discovered. The inability to obtain evidence does not imply it does not exist; consequently, it is assumed to exist with some probability. In the example, the evidentiary items B, H and L are assumed to be equally likely to be “true” or “false” (i.e., 50% probability).

Mathematical equations are set up that define the probability of each node of the BBN in terms of the probabilities of its parents. When the system of Bayesian equations associated with the nodes in Figure 5 is solved using a BBN tool (e.g., Netica), the probability of the evidence matching Misuse Case 1 is computed to be 89.6%. The high probability for Misuse Case 1 indicates a need to implement the requirements that arise from Misuse Case 1. However, it does raise the issue whether some or all of the requirements arising from Misuse Cases 2 and 3 should also be implemented. The answer is a technical as well as managerial decision, involving the level of risk and the economics of the situation, which are outside the scope of this paper.

Note that in the example in Figure 5, based on the forensic evidence collected, there is also a relatively high probability (56.3%), of dealing with another, as yet undefined, attribute list (which, in turn could refer

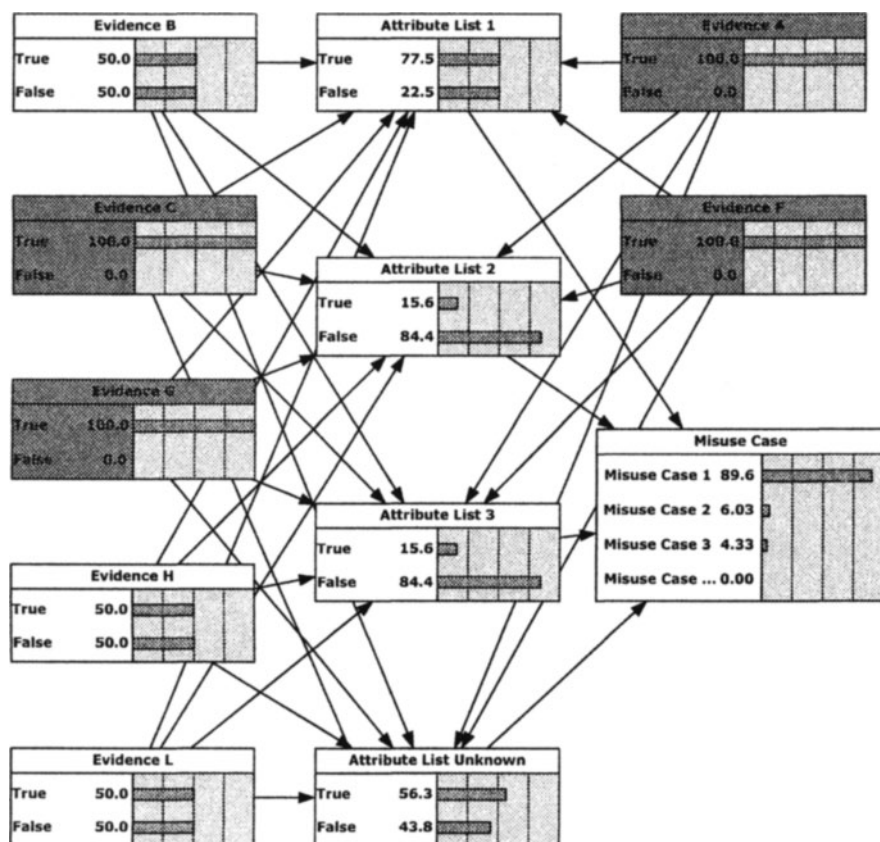


Figure 5. Probabilistic process using a Bayesian belief network.

to an unidentified misuse case and its associated security requirements). Additional forensic evidence is needed to confirm or deny the existence of undefined attribute list(s) and misuse cases.

7. Conclusions

Securing the railroad infrastructure is a high priority. Regulatory [6] and industry initiatives [11] related to the deployment of PTC systems have significantly increased railroad safety, but have also increased vulnerabilities [3]. A recent National Research Council (NRC) and National Security Telecommunications Advisory Committee (NSTAC) study [19] emphasizes that attacks on wireless networks can result in significant system degradation or disruption. PTC systems, because of their reliance on wireless networks, are prime targets for attack [13]. However,

previous work [2, 4, 5], while confirming the need to secure PTC systems and investigate security breaches, has not provided specific security requirements or developed decision processes for determining investigative scenarios [26].

Digital forensics has traditionally focused on electronic evidence gathered from computer systems and networks for use in legal proceedings. While considerable research has focused on using decision support systems to reason about evidence [7, 15, 22] and on generating crime scenarios from evidence using compositional reasoning [16], we believe that our use of use-misuse cases to determine the forensic evidence that should be collected for determining safety and security requirements is unique. The systematic analysis of forensic evidence from misuse cases proposed in this work will not only support accident investigations and contribute to the identification and prosecution of attackers, but will also increase the resilience of PTC systems to attack.

References

- [1] A. Bogen and D. Dampier, Preparing for large scale investigations with case domain modeling, *Proceedings of the Digital Forensic Research Workshop*, 2005.
- [2] A. Carlton, D. Frincke and M. Laude, Railway security issues: A survey of developing railway technology, *Proceedings of the International Conference on Computer, Communications and Control Technology*, pp. 1-6, 2003.
- [3] C. Chittester and Y. Haimen, Risks of terrorism to information technology and to critical interdependent infrastructures, *Journal of Homeland Security and Emergency Management*, vol. 1(4), 2004.
- [4] P. Craven, A brief look at railroad communication vulnerabilities, *Proceedings of the Seventh IEEE International Conference on Intelligent Transportation Systems*, pp. 345-349, 2004.
- [5] P. Craven and A. Craven, Security of ATCS wireless railway communications, *Proceedings of the IEEE/ASME Joint Rail Conference*, 2005.
- [6] Department of Transportation, 49 CFR Parts 209, 234 and 236: Standards for the Development and Use of Processor Based Signal and Train Control Systems – Final Rule, Technical Report, Washington, DC, 2005.
- [7] B. Falkenhainer and K. Forbus, Compositional modeling: Finding the right model for the job, *Artificial Intelligence*, vol. 51(1-3), pp. 95-143, 1991.

- [8] Federal Railroad Administration, Railroad Communications and Train Control, Technical Report, Department of Transportation, Washington, DC, 1994.
- [9] Federal Railroad Administration, Implementation of Positive Train Control Systems, Technical Report, Department of Transportation, Washington, DC, 1999.
- [10] Federal Railroad Administration, Benefits and Costs of Positive Train Control, Report in Response to the Request of Appropriations Committees, Department of Transportation, Washington, DC, August 2000.
- [11] Federal Railroad Administration, Positive Train Control, Technical Report, Department of Transportation, Washington, DC (www.fra.dot.gov/us/content/1265), 2003.
- [12] A. Gelman, J. Carlin, H. Stern and D. Rubin, *Bayesian Data Analysis*, Chapman and Hall/CRC, Boca Raton, Florida, 2003.
- [13] General Accounting Office, Critical infrastructure protection challenges and efforts to secure control systems, GAO Testimony before the Subcommittee on Technology Information Policy, Intergovernmental Relations and the Census, House Committee on Government Reform, House of Representatives, Washington, DC, March 2004.
- [14] F. Jensen, *Bayesian Networks and Decision Graphs*, Springer, Heidelberg, Germany, 2001
- [15] J. Keppens and Q. Shen, On compositional modeling, *Knowledge Engineering Review*, vol. 16(2), pp. 157-200, 2001.
- [16] J. Keppens and J. Zeleznikow, A model based reasoning approach for generating plausible crime scenarios from evidence, *Proceedings of the Ninth International Conference on Artificial Intelligence and Law*, pp. 51-59, 2003.
- [17] C. Lerman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, Prentice Hall, Upper Saddle River, New Jersey, 1998.
- [18] National Transportation Safety Board, Collision of Amtrak Train No. 90 and MARC Train No. 436, Railroad Accident Brief DCA-02-FR-010, Washington, DC (www.nts.gov/publictn/2003/RAB0301.htm), May 12, 2003.
- [19] Office of the President, The President's National Security Telecommunications Advisory Committee (NSTAC) Wireless Task Force Report, Washington, DC, January 2003.

- [20] S. Overmyer, B. Lavoie and O. Rambow, Conceptual modeling through linguistic analysis using LIDA, *Proceedings of the Twenty-Third International Conference on Software Engineering*, pp. 401-410, 2001.
- [21] J. Pauli and D. Xu, Threat-driven architectural design of secure information systems, *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pp. 136-143, 2005.
- [22] H. Prakken, Modeling reasoning about evidence in legal procedure, *Proceedings of the Eighth International Conference on Artificial Intelligence and Law*, pp. 119-128, 2001.
- [23] J. Reason, *Human Error*, Cambridge University Press, Cambridge, United Kingdom, 1990.
- [24] G. Sindre and A. Opdahl, Capturing security requirements through misuse cases, *Proceedings of the Ninth Norwegian Informatics Conference* (www.nik.no/2001/21-sindre.pdf), 2001.
- [25] G. Sindre and A. Opdahl, Templates for misuse case description, *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundations of Software Quality* (www.nik.no/2001/21-sindre.pdf), 2001.
- [26] G. Wimmel, J. Jurgens and G. Popp, Use case oriented development of security critical systems, *Information Security Bulletin*, vol. 2, pp. 55-60, 2003.

Chapter 13

MAC OS X FORENSICS

Philip Craiger and Paul Burke

Abstract This paper describes procedures for conducting forensic examinations of Apple Macs running Mac OS X. The target disk mode is used to create a forensic duplicate of a Mac hard drive and preview it. Procedures are discussed for recovering evidence from allocated space, unallocated space, slack space and virtual memory. Furthermore, procedures are described for recovering trace evidence from Mac OS X default email, web browser and instant messaging applications, as well as evidence pertaining to commands executed from a terminal.

Keywords: Macintosh computers, Mac OS X forensics

1. Introduction

Since its introduction in 1984, the Apple Macintosh has enjoyed a small, albeit vocal, user base. Nevertheless, it is surprising that very little has been published regarding forensic examinations of Macintosh computers.

This paper describes procedures for conducting forensic examinations of Apple Macs running Mac OS X. Due to space limitations, certain assumptions are made to limit the scope of our coverage. These assumptions are: (i) The forensic computer and the suspect's computer run version 10.4.3 of Mac OS X, the latest version as of November 2005; (ii) the suspect has not set the Open Firmware password (Open Firmware is a processor and system-independent boot firmware used by PowerPC-based Macs, analogous to the x86 PC BIOS); (iii) the suspect has not used encryption via the Mac OS X FileVault, a virtual volume encrypted with 128-bit AES; and (iv) the suspect's hard drive is formatted with the Hierarchical File System Plus, commonly referred to as HFS+, the default file system since Mac OS X's release in 2000.

Please use the following format when citing this chapter:

Craiger, P., Burke, P., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 159–170.

2. Mac OS X Background

Mac OS X is the successor to the original Apple Macintosh operating system that debuted in 1984. It has an entirely different code base from the original Mac OS, and is partially based on the NeXTSTEP operating system code base. Mac OS X is a UNIX-based operating system that comprises a FreeBSD-based subsystem and a Mach 3.0 microkernel. Although Mac OS X has a tightly integrated user interface that is very “Apple-like,” the underlying architecture is UNIX, with all the services and a command line interface that constitute the heart of UNIX operating systems.

3. Forensic Examination Procedures

Mac OS X provides a novel method for creating a forensic duplicate that requires placing the suspect’s computer into *target disk mode*. This mode allows an examiner to create a forensic duplicate of the suspect’s hard drive using a FireWire cable connecting the two computers. Target disk mode works with any version of Mac OS X or OS 8/OS 9 (predecessors to Mac OS X) with FireWire software version 2.3.3 or later [1].

Additionally, target disk mode supports an onsite preview of the contents of the suspect’s hard drive(s). Onsite previews are used when law enforcement is interested in seizing a computer only if evidence of probative value exists on the hard drive. An onsite preview allows an agent to conduct a search at the scene to determine if evidence exists to warrant the seizure of the suspect’s computer.

3.1 Creating a Forensic Duplicate

It is crucial that nothing causes the suspect’s hard drive to mount in read/write mode. This is because the process of mounting in this mode can cause changes to numerous files on the hard drive. We have discovered that booting a Windows 98 system causes changes to more than 400 files; approximately 400 files are modified in response to a graceful shutdown as well. **Disk arbitration**, the service that controls automatic disk mounting, must be disabled on the forensic computer prior to connecting the forensic and suspect computers. Under Mac OS X this service is performed by the file:

```
/usr/sbin/diskarbitrationd
```

Disk arbitration can be disabled by: (i) moving the file from its directory and rebooting the forensic computer, or (ii) moving its preference file:

```
/etc/mach.init.d/diskarbitrationd.plist
```

to a different location and rebooting. After reboot, an examiner can check the forensic computer's system log, or list the running processes and search for the keyword "diskarbitrationd," to verify that disk arbitration is not running. Note that this method only works for Mac OS X 10.4 or later versions. Earlier versions require a different method for disabling disk arbitration. Readers are referred to [4] for a discussion of procedures for disabling automounting on systems running OS X versions earlier than 10.4.

The next step is to connect the forensic computer and the suspect's computer with a FireWire cable, and then power on the suspect's computer while holding down the "T" key on the suspect's computer keyboard. This brings the suspect's computer into target disk mode. If the operation is successful, a FireWire icon is displayed on the suspect's computer monitor.

```
$ sudo ioreg -c "IOMedia" | grep 'FireWire Target Media' -A 10
| | | | | +-o AAPL FireWire Target Media ...
| | | | | | {
| | | | | | "Leaf" = No
| | | | | | "Writable" = Yes
| | | | | | "BSD Minor" = 3
| | | | | | "Preferred Block Size" = 512
| | | | | | "BSD Major" = 14
| | | | | | "BSD Name" = "disk1"
| | | | | | "Size" = 80026361856
| | | | | | "Content Hint" = ""
| | | | | | "Removable" = No
```

Figure 1. Device mapping information.

Next, it is necessary to identify the device file that maps to the suspect's hard drive: in UNIX-based systems, attached physical devices are represented via files located under the `/dev` directory. The Mac OS X utility `ioreg` is used to display the Input/Output registry for the Mac to determine which device files in the `/dev` directory are mapped to the suspect's hard drive. Within the output, it is necessary to search for the term "FireWire Target Media," as demonstrated in Figure 1.

Figure 1 indicates that the suspect's hard drive is mapped to `disk1`. The UNIX utility `dd` may now be used to create a forensic duplicate of the physical disk. The integrity of the duplicate can be verified using MD5 hashing:

```
$ dd if=/dev/disk1 of=./evidence.dd
$ md5 /dev/disk1 ./evidence.dd
MD5 (/dev/disk1) = 1cd57cb5d2e438d5be4bca9cf1664986
MD5 (evidence.dd) = 1cd57cb5d2e438d5be4bca9cf1664986
```

The duplicate may now be imported into any forensic software that understands the raw `dd` format to begin a forensic examination.

3.2 Previewing a Hard Drive

Information about the suspect's hard drive is needed in order to preview its contents. Like other BSD-based systems, volumes (partitions) are represented by the nomenclature `/dev/disk{n}s{m}`, where `{n}` is a number that denotes the physical drive, and `s{m}` represents the slice number `{m}`. Slice is BSD nomenclature for a volume. This information can be determined using the `hdiutil` utility as shown in Figure 2.

```
$ sudo hdiutil pmap /dev/disk1
Partition List
```

##	Dev_____	Type_____	Name_____	Start___	Size_____	End_____
0	disk1s1	Apple_partition_map	Apple	1	63	63
1		Apple_Free		64	262144	262207
2	disk1s3	Apple_HFS	Apple_HFS	262208	105406472	105668679
3		Apple_Free		105668680	262144	105930823
4	disk1s5	Apple_HFS	Apple_HFS	105930824	50370648	156301471
5		Apple_Free		156301472	16	156301487

Figure 2. Volume/partition information.

Figure 2 shows that the suspect's hard drive contains three volumes: `s1`, `s3` and `s5`. An onsite preview can be conducted by mounting one of the volumes as read-only and viewing the contents through a terminal. Before mounting the volumes, a directory (`evidence.s3`) is created on the forensic computer and the volume is manually mounted as follows:

```
$ sudo mount -t hfs -r /dev/disk1s3 evidence.s3/
```

Note that `-t hfs` specifies the type of file system (hierarchical file system, which is the Mac OS X default), and the `-r` flag indicates to mount the volume read-only. Files on the suspect's hard drive are viewed from the command line by changing to:

```
/Volumes/evidence.s3
```

on the forensic computer.

Files from the suspect's hard drive may be copied to the forensic computer without fear of causing any changes on the suspect's hard drive because it is mounted in read-only mode. We recommend using the command `cp -p` to copy files to the forensic computer because it maintains the original metadata (modified, accessed, changed timestamps, owner and group, permissions, etc.).

An onsite preview provides a view of files in allocated space only. The hard drive must be accessed at a physical level to access deleted files (in unallocated space) and slack space. Procedures for recovering evidence from unallocated space and slack space are described in the next section.

Target disk mode provides a very simple imaging and previewing solution for Macs. A second alternative not discussed here is the use of a bootable Linux CD (PowerPC version for pre-2006 Macs, or a x86 version as of January 2006) to create a forensic duplicate [6].

4. Recovering Deleted Files

A common first procedure performed during a forensic examination is to recover files from the Trash and deleted files; deleted files require access to the media at a physical level. Below we describe procedures for recovering deleted files in allocated, unallocated and slack space.

As with versions of Microsoft Windows, there are several methods to delete files under Mac OS X. From the desktop, a user can drag-and-drop a file onto the Trash icon. An alternative is to CTRL-Click the mouse over the file, which brings up a menu from which the *Move to Trash* option can be chosen. Both methods are analogous to dragging and dropping a file into the Recycle Bin in Microsoft Windows [9].

Similar to the behavior of the Windows Recycle Bin, files placed in the Trash are not deleted. Rather, the files are copied to a special hidden folder in the user's default directory, and deleted from their original locations. Thus, a copy of the file moved to the Trash exists in allocated space, and can be recovered by opening the Trash icon and moving the file from the Trash. Of course, the original deleted file resides in unallocated space.

The Trash is represented on the file system as a hidden folder, `.Trash`, under each user's home folder, as shown below:

```
~/Trash pc$ ls -al
total 160
drwx-----  5 pc  pc    170 Feb  4 11:49 .
drwxr-xr-x  47 pc  pc   1598 Feb  4 11:49 ..
-rw-r--r--   1 pc  pc   73028 Feb  3 16:40 Picture.1.png
-rwxr-xr-x   1 pc  pc    780 Feb  4 11:48 automount.sh
```

Note that the `.Trash` folder contains two deleted files that can be recovered by copying each file to the forensic computer. The modified, accessed and changed timestamps for the deleted files can be determined using the `stat -x` command as follows:

```
~/Trash pc$ stat -x Picture.1.png
File: "Picture.1.png"
Size: 73028      FileType: Regular File
Mode: (0644/-rw-r--r--)  Uid: ( 501/ pc)  Gid: ( 501/ pc)
Device: 14,2    Inode: 1454786    Links: 1
Access: Sat Feb  4 09:51:05 2006
Modify: Fri Feb  3 16:40:44 2006
Change: Fri Feb  3 16:40:44 2006
```

From a forensic standpoint it is important to note that dragging and dropping a file to the Trash does not change the file's date and time stamps. This enables an examiner to determine the original modified, accessed and changed timestamps for the file. However, it does not allow an examiner to determine when the file was placed in the Trash. In contrast, a file deleted from the command line is deleted in the traditional sense (explained below), bypassing the Trash.

When the Trash is emptied, the files are marked as deleted in the HFS+ catalog special file (analogous to the Master File Table in NTFS or the root directory in FAT-based file systems), and the blocks allocated in the allocation special file (analogous to the allocation bitmap file in NTFS or the FAT in FAT-based file systems [2]) are zeroed. However, the contents of the file remain until they are overwritten by the operating system.

An examiner must access the suspect's hard drive at a physical level to recover deleted files residing in unallocated space. A physical analysis views media without regard to the imposed file system. Deleted files can be recovered manually if the starting block and the size of the file are known. An examiner can identify a file's starting block by searching the media at a physical level, e.g., with a hex editor, and determining the beginning of the file by identifying specific keywords or a file signature. To demonstrate this, suppose that a deleted file's starting block is 4355500, and the file comprises 18 contiguous blocks. Given that the default block size under Mac OS X is eight sectors per block (512 bytes/sector $\times$ 8 = 4096 bytes/block), the deleted file can be recovered as follows:

```
$ dd if=/dev/disk1 of=./evidence bs=4096 skip=4355500 count=18
```

The block size is set to the default block size used by HFS+ (**bs=4096**) and 18 contiguous blocks (the number of blocks required to store the file) are recovered from the starting block of the file (**skip=4355500**) onwards. This procedure recovers all the data up to the end of the last sector of the block. Therefore, the procedure will also recover the slack space, i.e., unallocated space located after the end-of-file marker, up to the end of the last block of the file, unless the file size is a multiple of the block size, meaning there is no slack space. Note also that the procedure

will recover the entire contents of a deleted file only if the file was not fragmented.

Foremost [12] is a command line utility that automates the recovery of evidence from unallocated space. It recovers evidence from dd images and several other image formats based on internal data structures or file signatures such as headers and footers. Foremost is open source, and distributed as source code that can be compiled under Mac OS X.

4.1 Recovering Evidence from Virtual Memory

Even when a file has been deleted and physically wiped using a secure deletion utility [5], there may be traces of the file in virtual memory. It is possible to recover the evidence by searching through virtual memory, which is represented as a file called `swapfile` located in the directory `/var/vm`.

```
$ ls -al /var/vm
total 131072
drwxr-xr-x    4 root  wheel      136 Oct 14 10:50 .
drwxr-xr-x   24 root  wheel      816 Oct 14 10:52 ..
drwx--x--x   18 root  wheel      612 Oct 11 11:20 app_profile
-rw-----T    1 root  wheel 67108864 Oct 14 10:50 swapfile0
```

UNIX utilities can be used to search for keywords within `swapfile`. The command `strings -o` is executed on `swapfile` to extract human-readable content in the 7-bit ASCII range. Next, `grep` is used to search for an appropriate keyword as shown below.

```
$ sudo strings -o /var/vm/swapfile0 | grep cyberterror -C 2
34649683 Definitions of cybert#849CA.doc
34649751 PD+W8BNMSWD
34649786 cyberterrorism
34649847 KLittleBuddy:Users:pc:Desktop:cyberterrorism...
34650027 AUsers/pc/Desktop/cyberterrorism.doc
34650667 BNMSWD
34650707 craiger.pollitt.ch#FFFFFFFF.doc
```

Generally, this procedure is more successful at recovering textual information as opposed to binary or graphical data. Unfortunately, the BSD version of `strings` does not support searches for text encoded in UNICODE. However, Sleuth Kit [11] includes a utility called `srch_strings` that supports searches for text encoded in 16-bit or 32-bit UNICODE (bigendian or littleendian). Sleuth Kit is distributed as source code that can be compiled for Mac OS X. The `srch_strings` utility can be used to search `swapfile` for a UNICODE encoded keyword as demonstrated below.

```
/var/vm pc$ sudo srch_strings -e 1 swapfile0 | grep -in 'defense'
79:During the 1960s, the Department of Defense
```

5. Recovering Application-Related Evidence

Mac OS X includes an email reader (Apple Mail), a web browser (Safari), and an instant messaging application (iChat), all of which leave trace evidence on a hard drive. Knowing the location of this trace evidence and the format of the evidence are crucial during an onsite preview and when conducting a logical analysis in the laboratory. Below we describe trace evidence locations for each of these applications, as well as any special reformatting that must occur to make the evidence human readable. Additionally, we demonstrate the recovery of evidence from the UNIX command line, assuming that the suspect used the FreeBSD subsystem in the commission of a crime.

5.1 Apple Mail

Apple Mail is a full-featured email application that supports multiple POP3 and IMAP accounts and advanced filtering. User email is stored in the directory:

```
/Users/<username>/Library/Mail
```

The `/Users` directory is the Mac analog of the UNIX `/home` directory, where user files are stored. Consistent with the UNIX philosophy for a multi-user system, each user has his/her own directory.

Apple Mail files were stored in `mbox` format prior to Mac OS X 10.4. The `mbox` files are simple flat text files with individual emails appended to the end of the file [10]. As of Mac OS 10.4, Apple changed the default format to `emlx`, where each email is in its own file in ASCII format. Apparently, the change from `mbox` to `emlx` was made to allow for more thorough indexing under Apple's Spotlight integrated search technology [3]. Because each email is a simple text file, UNIX utilities can be used to search for specific keywords within the files.

One alternative is to convert the `emlx` files to `mbox` format, and then import the `mbox` file into another (non Apple Mail) email application, e.g., Outlook, Thunderbird or Eudora. Unfortunately, it is not possible to directly import `emlx` files into a secondary mail application for viewing emails as these applications do not understand the `emlx` format.

5.2 Safari Web Browser

Apple's bundled web browser, Safari, has become the *de facto* standard browser for Mac users. Safari is gaining increasing market share as Internet Explorer is no longer bundled with the latest versions of Mac OS X.

A suspect's web browsing history, download history and bookmarks can be used as evidence in criminal cases. Under Mac OS X the files used to store this information are named `History.plist`, `Downloads.plist` and `Bookmarks.plist`, respectively, and are stored in:

`/Users/<username>/Library/Safari`

`History.plist` stores information about the web pages the user has visited along with recent web searches performed from Safari's Google search bar. `Bookmarks.plist` stores bookmarks and `Downloads.plist` stores the history of files downloaded from Safari. A forensic examiner can parse these files using the `defaults read` command. Below, the `History.plist` file is read to determine a suspect's web browsing history.

```
$ defaults read ~/Library/Safari/History
WebHistoryDates = (
"http://www.google.com/search?q=bomb+making&ie=UTF-8&oe=UTF-8";
lastVisitedDate = "153417328.9";
title = "bomb making - Google Search";
visitCount = 1;
---
"" = "http://slashdot.org/";
lastVisitedDate = "160765082.1";
title = "Slashdot: News for nerds, stuff that matters";
visitCount = 3;
```

Note that the full path to the file must be included (the `.plist` extension must be excluded) to parse the file properly. The snippet above indicates that the suspect performed a Google search for the keywords "bomb making," and visited Slashdot three times.

Safari cache files are located in:

`/Users/<username>/Library/Caches/Safari`

Below is a directory listing that displays the naming scheme used:

```
$ ~/Library/Caches/Safari/00/00 $ ls -al
total 256
drwx----- 13 pc  pc    442 Aug 21 17:05 .
drwx----- 18 pc  pc    612 Mar  6 2005 ..
-rw-----  1 pc  pc   1385 Jul 27 20:04 1113936647-3722688267.cache
-rw-----  1 pc  pc  48227 Aug 21 17:05 1199711745-0794204939.cache
```

The browser cache is separated into several files that span a set of two-digit numbered folders, each comprising another set of two-digit numbered folders. The files can be viewed by copying these folders to the forensic computer and opening them in Safari.

Safari cookie files are stored in the file:

```
/Users/<username>/Library/Cookies/Cookies.plist
```

As with the other `.plist` files, it is possible to transfer these files from the suspect's computer to the forensic computer and use Safari to view them by selecting *Preferences:Security:Show Cookies* from the menu.

5.3 iChat Instant Messaging

Instant Messaging (IM) applications are bundled with all major operating systems. Additionally, all major web-based mail providers, including AOL, MSN, Yahoo! and Gmail provide instant messaging services.

Different IM applications store trace evidence pertaining to instant message conversations differently. Some may store instant messages on servers only, whereas others may keep a history of previous conversations by default. iChat does not automatically store previous conversations; however, a preference allows users to store iChat conversations on the local machine. The default location for these stored iChats is:

```
/Users/<username>/Documents/iChats
```

Individual iChat sessions are named as follows:

```
<username> on <date> at <time>.ichat
```

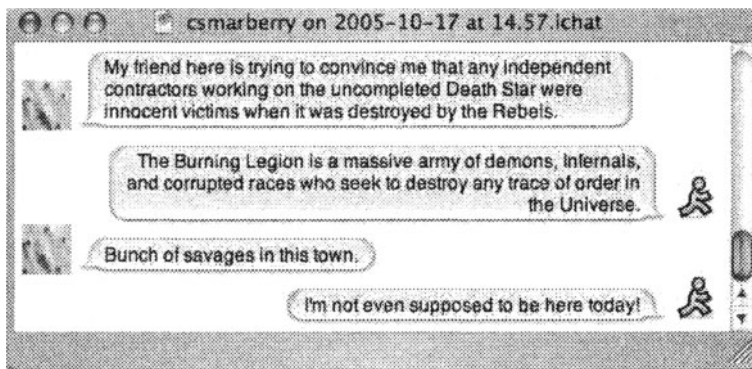


Figure 3. iChat

It is possible to view the contents of a session by copying a file to the forensic computer and opening the file by double clicking. Figure 3 shows a recovered iChat conversation between two users.

5.4 Command Line Input

A savvy criminal may open a terminal and use UNIX commands in the commission of a crime. This is most likely where an examiner will find evidence of a network intrusion.

The **bash** shell is the default shell for Mac OS X 10.4. It records commands executed by a user from the command line to a file named **.bash.history**, which is a hidden file in the user's home directory. To illustrate this, we copied the **.bash.history** file from the suspect's computer to the forensic computer. The last few entries in this file are:

```
$ tail .bash_history
...
sudo nmap -sS 192.168.1.0/24 > /Volumes/leet/recon.txt
cd /Volumes/leet/
less recon.txt
rm recon.txt
```

According to these entries, the suspect performed a port scan on an internal network (note the private addresses). The suspect saved the results to another volume named **leet**, viewed the file, and then deleted it. Unless this file has been overwritten, it can be recovered by accessing the **dd** image at the physical level, or accessing the associated **/dev** device in the target disk mode, and searching for keywords that we know existed in the deleted file:

```
$ cat ./evidence.dd | strings | grep -i 'nmap' -C 10
Starting nmap 3.81 ... at 2005-11-11 14:43 EST
Interesting ports on 192.168.1.2:
(The 1657 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
515/tcp    open  printer
548/tcp    open  afpovertcp
3689/tcp   open  rendezvous
5000/tcp    open  UPnP
```

Space limitations preclude us from discussing all the possible locations in a UNIX-based system from which evidence may be recovered. Interested readers are referred to [7, 8] for additional details.

6. Conclusions

Mac OS X forensics is an important but relatively unexplored area of research. This paper has discussed procedures for recovering evidence from allocated space, unallocated space, slack space and virtual memory, as well as Mac OS X default email, web browser and instant messaging applications, and command line input. Due to space limitations,

it was not possible to cover other well-known applications, including email readers (e.g., Microsoft Entourage and Mozilla Thunderbird), web browsers (e.g., Microsoft Internet Explorer and Mozilla Firefox), and popular instant messaging applications (e.g., AOL AIM and Adium). It is important that forensic examiners become familiar with these applications and the locations of associated trace evidence. Examiners must also have a thorough understanding of the various versions of Mac OS X, some of which require specialized forensic procedures.

References

- [1] Apple Computer, How to use FireWire target disk mode (docs.info.apple.com/article.html?artnum=58583), 2002.
- [2] Apple Computer, Technical Note TN1150: HFS Plus Volume Format (developer.apple.com/technotes/tn/tn1150.html), 2004.
- [3] Apple Computer, Working with Spotlight (developer.apple.com/macosx/spotlight.html), 2005.
- [4] BlackBag Tech, FireWire target disk mode guidelines (blackbagtech.com/images/BBT_FireWire.Target.Mode.pdf), 2004.
- [5] P. Burke and P. Craiger, Assessing trace evidence left by secure deletion programs, in *Advances in Digital Forensics II*, M. Olivier and S. Shenoi (Eds.), Springer, New York, pp. 185-195, 2006.
- [6] P. Craiger, Recovering evidence from a Linux system, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), New York, pp. 233-244, 2005.
- [7] D. Farmer and W. Venema, *Forensic Discovery*, Prentice-Hall, Upper Saddle River, New Jersey, 2004.
- [8] K. Jones, R. Bejtlich and C. Rose, *Real Digital Forensics: Computer Security and Incident Response*, Addison-Wesley Professional, New York, 2005.
- [9] Microsoft Corporation, How the recycle bin stores files (support.microsoft.com/default.aspx?scid=kb;en-us;136517), 2004.
- [10] Network Working Group, RFC 4155 – The Application/Mbox Media Type (www.faqs.org/rfcs/rfc4155.html), 2005.
- [11] Sleuthkit.org, Sleuth Kit (www.sleuthkit.org).
- [12] Sourceforge.net, Foremost (foremost.sourceforge.net).

Chapter 14

DETECTING DATA CONCEALMENT PROGRAMS USING PASSIVE FILE SYSTEM ANALYSIS

M. Davis, R. Kennedy, K. Pyles, A. Strickler and S. Sheno

Abstract Individuals who wish to avoid leaving evidence on computers and networks often use programs that conceal data from conventional digital forensic tools. This paper discusses the application of passive file system analysis techniques to detect trace evidence left by data concealment programs. In addition, it describes the design and operation of Seraph, a tool that determines whether certain encryption, steganography and erasing programs were used to hide or destroy data.

Keywords: Data concealment programs, trace evidence, program detection

1. Introduction

Encryption, steganography and erasing tools are increasingly used by malicious individuals to hinder forensic investigations [14]. The 2005 indictment of former *Newsday* CEO Robert Johnson in U.S. District Court [22, 23] exemplifies the importance of detecting the presence of data concealment programs on seized media. Johnson was charged with the receipt and possession of child pornography and obstruction of justice for destroying digital evidence. According to the indictment, he erased thousands of illegal pornographic images using Evidence Eliminator, an erasing program that was found on his computer.

Current methods for detecting data concealment programs are resource intensive and time consuming. In particular, they attempt to discover hidden data without first verifying whether or not data is actually concealed [5, 11, 16]. Forensic tools, e.g., FTK, employ known file filters to alert forensic investigators to the presence of certain utilities on a seized computer [16]. However, they do not provide information if

Please use the following format when citing this chapter:

Davis, M., Kennedy, R., Pyles, K., Strickler, A., Sheno, S., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 171–183.

a specific utility was installed, executed and subsequently uninstalled. Obviously, such evidence could be very valuable in legal proceedings.

Forensic investigators need tools to identify the specific programs and techniques used to hide or erase data on a seized computer. Having made this determination, attempts can be made to recover the data by exploiting known vulnerabilities in the programs and their data concealment techniques. This paper discusses the use of passive file system analysis techniques to detect trace evidence left by encryption, steganography and erasing programs. In addition, it describes the design and operation of Seraph, a tool that assists forensic investigators in identifying the programs used to hide or destroy data.

2. Data Concealment Programs

This section discusses data concealment involving the use of encryption, steganography and erasing tools.

2.1 Encryption Tools

Encryption tools pose unique challenges for law enforcement [19]. Such tools are increasingly used to hide legal – and illegal – information [3]. Some encryption programs, e.g., BestCrypt [21] and Cryptainer [6], create a secure container or vault on an allocated area of a hard drive. Files stored in this container are accessed via a key that is usually a password known to the user.

Another approach is to encrypt each file individually. Tools such as Folder Lock [17] and Kryptel [13] allow users to choose from a list of popular encryption algorithms and enable them to hide the encrypted files using advanced file manipulation techniques. Other software suites, e.g., Microsoft Office, provide application-level encryption, allowing users to password-protect sensitive information using various techniques, e.g., RC4-based encryption.

AccessData's Password Recovery Toolkit (PRTK) is often used by law enforcement agents to recover passwords and access evidence stored in encrypted files [1, 14, 16]. PRTK employs three methods for recovering passwords: algorithm attacks, key space attacks and dictionary attacks. Files encrypted with application-based encryption are usually easy to break with PRTK, but those using file system or vault encryption are more difficult. Using PRTK is very time consuming. To address this problem, AccessData has created a companion utility, Distributed Network Attack (DNA), that engages multiple computers on a network to break encryption. However, smaller law enforcement agencies do not have the network resources to use DNA effectively.

The process of breaking encryption may be expedited by first identifying the specific encryption program that was used. Software utilities such as Find Protected [2] and Encrypted File Search [8] find files encrypted with specific protocols and programs. However, these utilities examine every file on a system to detect the presence of encrypted data; this is a very time-consuming process. A better approach is to identify the specific software tool used for encryption, and exploit its vulnerabilities to decrypt data.

2.2 Steganography Tools

Steganography is the process by which a message is hidden so that only the sender and the intended recipient know of its existence. Steganography tools typically employ a graphic file (e.g., bitmap or jpeg file) as a carrier for hidden data. A variety of algorithms and techniques can be used to hide data within a carrier file without making noticeable changes to the file [5, 24].

Programs that implement steganography are widely available and are increasingly used to conceal evidence of illegal activities. Several software tools are available for analyzing graphic files for the presence of steganography [5, 11, 12, 25]. WetStone's Stego Suite [25], one of the premier steganalysis tools, analyzes hard drives for evidence of known steganography algorithms. However, Stego Suite (and other tools) analyze all the files on a suspect hard drive, which is a resource-intensive operation.

2.3 Erasing Tools

Erasing tools are designed to destroy data, ideally rendering it unreadable by digital forensic tools. One popular erasing tool is Internet Eraser Pro [9], which eliminates all traces of Internet use; it deletes cookies, clears the history and wipes temporary Internet files. Such tools are often used on shared computers, for example, to hide evidence of visits to pornographic websites.

Erasing tools, e.g., Evidence Eliminator [18], may be used to wipe selected files or entire drives. Indeed, it is difficult, if not impossible, to recover evidence after such tools have been used.

Techniques employed by erasing tools vary widely [4]. Some utilities overwrite the targeted files with random data and then delete them in the usual fashion. Others write random data and then zero the disk space. Still others zero the drive in multiple passes. Many tools only make one or two passes when wiping a drive while others surpass even the U.S. Department of Defense's erasing standard [7]. Since some tools

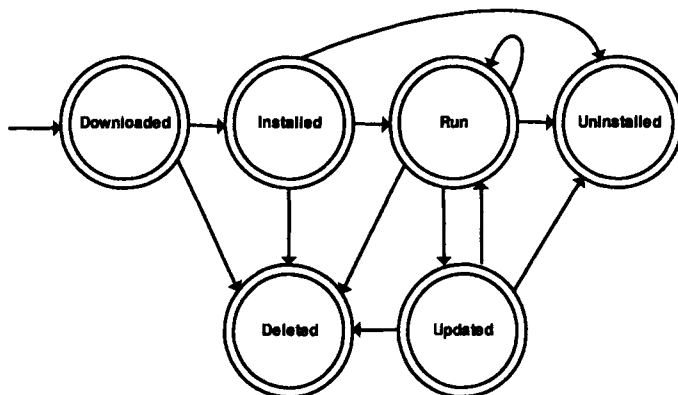


Figure 1. Program lifecycle.

employ weaker data destruction protocols than others, it is important to know the program or at least the protocol used to erase data [4].

The discovery and identification of erasing tools used on a computer can be extremely useful in criminal investigations. As in the Johnson case [22, 23], the presence of an erasing tool could indicate that incriminating evidence was destroyed and possibly lead to an obstruction of justice charge. Many erasing tools leave trace evidence of their activities [4]. Therefore, it may be possible to determine if an erasing tool was installed, and whether or not it was executed.

3. Program Lifecycle

The lifecycle of a software program follows a deterministic path (Figure 1). Information obtained by analyzing the lifecycles of data concealment tools can be extremely useful in forensic investigations. This is because the tools often leave trace evidence during various phases of their lifecycles, e.g., when they are installed, executed or uninstalled.

For our purposes, a program's lifecycle begins when it is downloaded from a network or a physical medium (Figure 1). Many programs include installation packages that copy the files to the system and change registry settings and configuration files. However, some programs must be copied manually by the user. Both these situations cause a program to move to the installed state.

After a program is installed, it may move to the executed state (i.e., when it is executed). However, a program can also be uninstalled without being executed.

After being executed, a program may be updated, in which case it moves to the updated state. The updated state is similar to the installed state, with the exception that the previous version of the program was

executed. A program in the updated state may move to the executed state or the uninstalled state.

A program moves to the uninstalled state when a utility is used to remove it from the system. The lifecycle of a program ends when it moves to the uninstalled state.

A program in the downloaded, installed, executed or updated states may move to the deleted state. A program is in the deleted state when it is manually removed from the system, whether or not an uninstallation package exists.

Other scenarios are possible that might alter the program lifecycle. For example, a corrupted program may require the reinstallation of a new version of the program. Most of these scenarios can be treated as the start of the lifecycle of a new program.

In general, the lifecycle of a program could halt in any of the aforementioned states. The state of a program can be determined by examining the evidence left on the system.

4. Potential Evidence

Practically every program generates information and leaves trace evidence during each stage of its lifecycle. Understanding the subtle changes to a file system produced by a specific program (e.g., an erasing tool) during its lifecycle and locating this trace evidence on a seized computer can enable an investigator to determine that the program was installed, executed or uninstalled on the computer.

Downloaded State: The primary source of evidence in this state is the actual downloaded program. The downloaded program is typically a compressed file (e.g., ZIP), or an executable installation package file. Also, the downloading process itself may leave trace evidence in the Internet history and log files.

Installed State: After a program has entered the installed state, files may be added to the hard drive. If the program was copied to a directory from a compressed file or downloaded directly, the only evidence is the program itself. If an installation package was used, evidence of the installation often exists in the registry settings and entries in shared configuration files [10].

Executed State: Once a program has entered the executed state, several traces of its activity can be found. During its first execution the program may ask the user for configuration information and create a file

11/06/2005	12:04 PM	95,028	VISIO.EXE-1F9B2047.pf
11/01/2005	01:49 PM	36,210	VPC32.EXE-29593AFF.pf
11/01/2005	01:49 PM	21,570	VPDM_LU.EXE-1D1611C8.pf
10/28/2005	02:15 PM	30,922	WCESMGR.EXE-2FB86E92.pf
11/02/2005	06:23 PM	53,510	WINLOGON.EXE-32C57D49.pf
11/03/2005	03:28 PM	90,390	WINRAR.EXE-39C6DAD9.pf
11/03/2005	03:57 PM	67,426	WINWORD.EXE-37F6AE09.pf

Figure 2. Windows XP Prefetch entries.

or add entries to the Windows registry. Some encryption programs create a container for the encrypted data. The presence of such a container would indicate that an encryption program was executed. A program could also create user files, e.g., Microsoft Word (.doc) files.

Another example is the Microsoft Windows XP Prefetch directory. This directory, located at %Windir%\Prefetch, contains information about executables and where they are stored on disk (Figure 2). The Prefetch capability, which is turned on by default, enables Windows XP to access files more efficiently. A program has an entry in the Prefetch directory only after it has been executed [20].

Trace evidence added to a file system when a program enters the executed state helps determine that a program was actually executed. This is more useful than knowing that the program was installed (and possibly never used).

Uninstalled State: A program in the uninstalled state has been removed from the system using an installation program or a separate utility. Little direct evidence remains as program files and associated folders are removed from the disk. However, shared Windows configuration files, e.g., win.ini and system.ini, as well as the Windows registry may still contain traces of the program. Also, the Windows Prefetch entry is typically retained.

Deleted State: Since the deleted state requires that the user manually delete the program and associated files, it is likely that most, if not all, the program files are removed. However, as with the uninstalled state, it is possible to find traces of the program in shared configuration files, registry entries and the Windows Prefetch directory. Users who manually delete files also tend to miss shared files, e.g., dynamic link libraries (DLLs) in the %Windir%\System directory.

Updated State: Evidence retained in the updated state is similar to that left in the installed and executed states. If a drive is seized when a program is in the updated state, evidence that the program was executed

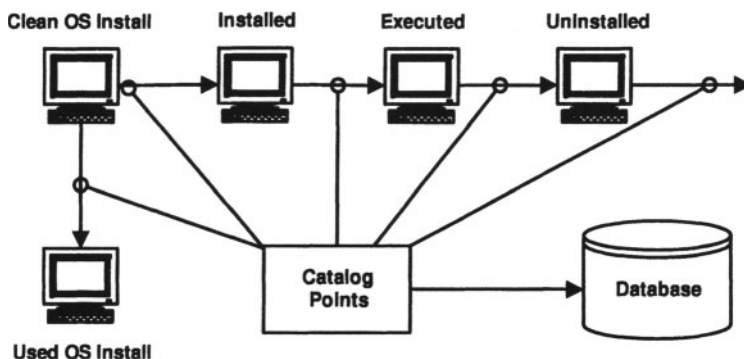


Figure 3. Fingerprint generation system.

prior to update should exist. However, this evidence is not from the execution of the current version of the program. When a program is updated, a completely new version may be installed in other folders in the file system; this is treated as a separate program. If the program itself or its data are updated, the behavior of the program is affected, and new evidence may be left in the executed state.

5. Generating Program Fingerprints

A laboratory configuration for generating program fingerprints during the various stages of its lifecycle is presented in Figure 3. One computer was configured with a base install of Windows XP without any other software packages. This computer was imaged and the image copied to other identical computers to create a baseline configuration for analysis. Several encryption, steganography and erasing programs were downloaded and burned on a single CD-ROM. Each program was analyzed independently.

After installing a program on one of the test machines, the machine was immediately switched off. This was done to limit file system activity performed by Windows upon shutdown. The hard drive was moved from the machine and connected to an analysis machine via a hardware write blocker. The drive was processed and cataloged using customized software. After this process was completed, the drive was replaced in the test machine and the program was executed, following which the drive was removed and re-analyzed. The drive was again returned to the test machine and the program uninstalled, upon which the drive was removed and analyzed one last time.

This procedure created the base set of data pertaining to the program during various stages in its lifecycle. Note that the updated and deleted

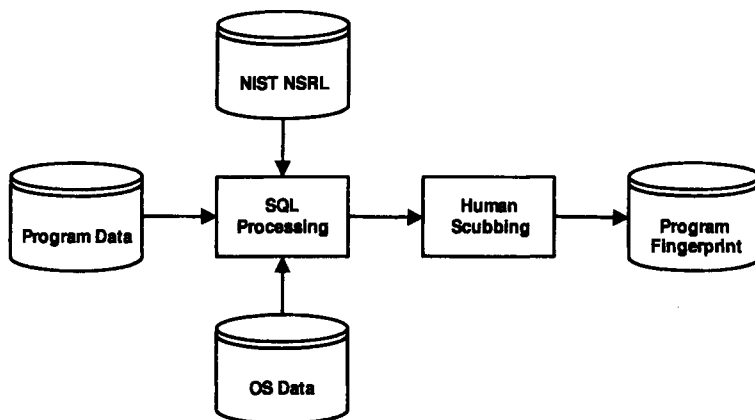


Figure 4. Generating program fingerprints.

states were not analyzed because programs typically generate little new evidence during these stages of their lifecycles.

Next, the test machines were re-imaged with the original base image and placed in a shared laboratory for general use for a period of two weeks. This allowed the base image to gather user and operating system changes that were used to reduce the program lifecycle data set. After the two-week period, the hard drives were removed from the machines and analyzed as before.

The images from the computers in the shared laboratory provided a large data set for programs in various stages of their lifecycles. These images were processed using a fingerprint generation engine to remove erroneous and redundant data, and to create fingerprints for each program during its lifecycle stages (Figure 4). The NIST National Software Reference Library [15], the clean base image, and the used base images were all used to process the data and create fingerprints. Each fingerprint was manually verified before producing the final program fingerprint.

Upon analyzing the program fingerprints, it was determined that many of the encryption, steganography and erasing programs shared files (e.g., DLLs and installation package files). This created a significant challenge to matching program fingerprints accurately. For example, in the uninstalled state, a program typically leaves a copy of the uninstall tool, which is shared by other programs. However, it was possible to determine which programs were present on the system using information such as full paths and registry settings.

Program	State	Matched Entries	Fingerprint Entries	Confidence	Category
Complete Cleanup	Installed	39	42	92.86%	ERASING
Complete Cleanup	Run	43	46	93.48%	ERASING
Eraser	Installed	41	48	85.42%	ERASING
Eraser	Run	40	49	81.63%	ERASING
Track Eraser	Installed	53	55	96.36%	ERASING
Track Eraser	Run	53	56	94.64%	ERASING

Figure 5. Results of Seraph fingerprint matching.

6. Seraph Program Detection Tool

A software tool named Seraph was created to assist forensic investigators in cataloging hard drives and detecting programs of interest based on their fingerprints. Seraph helps determine if any of the fingerprinted programs are/were present on the catalogued drive. Program fingerprints may be matched by filename, full path or MD5 hash of the file. Each of these options produces slightly different results. Filename matching produces a significant number of false positives. For example, many programs include a file named `readme.txt`. The number of false positives is reduced greatly when the full path information of files is considered, especially when the programs are installed in default directories.

Matching MD5 hash values can be effective, but it can actually reduce the number of features that a program will match against its fingerprint. This is because different versions of the same program (due to changes to executables and associated files) yield different MD5 hash values. However, programs typically use the same default installation folder and filenames, allowing filename and full path matching techniques to identify different versions of a program.

Seraph allows investigators to set a “confidence level” threshold for fingerprint matching. Many of the fingerprinted programs share DLLs or have files with identical names (e.g., `readme.txt`). By setting the threshold, an investigator can adjust the number of features used for fingerprint matching. When searching for uninstalled or deleted programs, a lower threshold should be used as many of the files associated with the programs have been removed. Since this can produce many false positives, the investigator must examine the results very carefully.

Once all the options have been selected, the investigator can use Seraph’s “detect” function to match programs on the image with the stored fingerprints (Figure 5). Within seconds, Seraph displays a list of matched fingerprints at the specified confidence level. The investigator can then examine the details of the fingerprint matches or create a text report (Figure 6).

```
Seraph Report
Created on: 11-06-2005
=====

Image Details
=====
Image Name: Test Image 3
Image Taken By: Amanda Strickler
Image Date: 9/26/2005 at 12:57:58
=====

Options
=====
Filename: Yes
Fullpath: Yes
Hash: No
Confidence Level: 30%
=====

Matched Programs
=====

Program: Complete Cleanup
State: Installed
Matched Entries: 39
Fingerprint Entries: 42
Confidence: 92.86%
Category: ERASING

Fullpath: \Documents and Settings\All Users\Start Menu\Programs\
Complete Cleanup Trial\Complete Cleanup Trial.lnk
Filename: Complete Cleanup Trial.lnk
Fingerprint File Size: 700
Fingerprint MD5 Hash: e898dbb32d5c7e79145f1b4a17321273
Image Size: 700
Image Hash: e6fca3fb2f5a8476e188bbbc4629dda7

Fullpath: \Documents and Settings\User\Desktop\Complete Cleanup
Trial.lnk
Filename: Complete Cleanup Trial.lnk
Fingerprint File Size: 688
Fingerprint MD5 Hash: 28be74c11e765b967d5661dc78f41134
Image Size: 688
Image Hash: 37d19c438019dbea9bde43e9bbc4b5c3

Fullpath: \Program Files\Complete Cleanup Trial\activex_t.htm
Filename: activex_t.htm
Fingerprint File Size: 1064
Fingerprint MD5 Hash: 7fa6f58a735db46d215c683bfc4b262d
Image Size: 1064
Image Hash: 7fa6f58a735db46d215c683bfc4b262d
```

Figure 6. Seraph report.

7. Results and Discussion

Seraph provides an investigator with results based on program name, state and confidence level threshold. In the example output in Figure 5, it is possible to conclude that the program Complete Cleanup may have been executed on the system. Since the fingerprint catalog for the executed state contains more entries than the installed state, it can be ascertained that some files are added during program execution. Furthermore, the investigated drive exhibits additional matches with the executed fingerprint. Seraph offers a "detailed analysis" option that displays all matched features. This can be used to determine if a Windows Prefetch entry or certain configuration files were added to the system, indicating that the program had entered the executed state.

Seraph was blind tested for accuracy. Three test hard drives were loaded with several data concealment programs at various stages of their lifecycles. The test drives were then imaged by Seraph. Using the default settings to search for filenames and full paths with a confidence threshold of 30%, Seraph correctly detected 7 out of 7 random test programs in the installed and executed states.

Upon decreasing the confidence threshold to 15%, Seraph correctly detected 3 of 3 programs in the uninstalled state. However, Seraph did return 8 false positives at the lower confidence level. Nevertheless, using Seraph's "detailed analysis" option, it was possible to determine which matches were false positives. Figure 5 shows the relevant portion of a Seraph report that can aid in the analysis of these results.

Seraph is being tested by detectives from the Tulsa Police Department's Cyber Crimes Unit and agents from the Oklahoma State Bureau of Investigation. Currently, Seraph is able to detect data concealment programs based on the existence of files. But Seraph's fingerprint entries also contain information about changes to shared configuration files and registry settings; the next version of Seraph will use this information to improve the accuracy of program detection. Furthermore, the new version will incorporate several automated features to reduce the amount of manual analysis performed by forensic investigators.

8. Conclusions

Trace evidence left by data concealment programs during various stages of their lifecycles can be analyzed to provide valuable information, including the names and versions of the programs, and whether they were installed, executed or uninstalled. Having established the presence of a specific data concealment program on a seized computer, an investigator can attempt to exploit known vulnerabilities in the pro-

gram to recover concealed or erased data. The Seraph program detection tool uses passive file system analysis to obtain information pertaining to the use of encryption, steganography and erasing programs. The current version has performed reasonably well in tests and in real investigations by two law enforcement agencies. However, more research and development efforts are needed to expand the coverage of the program detection tool and improve its accuracy.

References

- [1] Access Data, Password Recovery Toolkit (PRTK) (www.accessdata.com/products/prtk).
- [2] AKS-Labs, Find password protected files (www.aks-labs.com/solutions/find-password-protected.htm).
- [3] C. Brown, Detecting and collecting whole disk encryption media, presented at the *Department of Defense Cyber Crime Conference*, Palm Harbor, Florida, 2006.
- [4] P. Burke and P. Craiger, Assessing trace evidence left by secure deletion programs, in *Advances in Digital Forensics II*, M. Olivier and S. Shenoi (Eds.), Springer, New York, pp. 185-195, 2006.
- [5] K. Curran and K. Bailey, An evaluation of image-based steganography methods, *International Journal of Digital Evidence* vol. 2(2), 2003.
- [6] Cypherix, Cryptainer LE (www.cypherix.com/cryptainerle).
- [7] Defense Security Service, *National Industrial Security Program Operating Manual (NISPOM)*, DoD 5220.22-M, U.S. Department of Defense (www.dss.mil/isec/nispom.0195.pdf), 1995.
- [8] Free Downloads Center, Encrypted Files Search 1.2 (www.freedownloadscenter.com/Utilities/Misc_Encryption_Uutilities/Encrypted_Files_Search.html).
- [9] Giant Internet, Internet Eraser Pro (www.interneteraser.net).
- [10] InstallShield, Creating registry keys (helpnet.installshield.com/robo/projects/helpplibdevstudio9/IHelpRegistryKeys.htm).
- [11] J. Jackson, G. Gunsch, G. Lamont and R. Claypoole, Blind steganography detection using a computational immune system: A work in progress, *International Journal of Digital Evidence*, vol. 1(4), 2003.
- [12] G. Kessler, An overview of steganography for the computer forensics examiner, *Forensic Science Communications*, vol. 6(3), 2004.

- [13] Kryptel, Kryptel Encryption Suite (www.kryptel.com/products/kryptel).
- [14] K. Mandia, C. Proise and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.
- [15] National Institute of Standards and Technology (NIST), National Software Reference Library (NSRL), Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, Maryland (www.nsrl.nist.gov).
- [16] B. Nelson, A. Phillips, F. Enfinger and C. Steuart, *Guide to Computer Forensics and Investigations*, Thompson Course Technology, Boston, Massachusetts, 2004.
- [17] NewSoftwares.net, Folder Lock (www.newsoftwares.net/folderlock).
- [18] Robin Hood Software, Evidence Eliminator (www.evidence-eliminator.com/product.d2w).
- [19] J. Seigfried, C. Siedsma, B. Countryman and C. Hosmer, Examining the encryption threat, *International Journal of Digital Evidence*, vol. 2(3), 2004.
- [20] J. Sheesley, Use XP's Prefetch feature to improve system performance, TechRepublic (techrepublic.com.com/5100-1035_11-5165773.html?tag=e064#), 2004.
- [21] Softpedia, Bestcrypt 7.20.2 (www.softpedia.com/get/Security/Encrypting/BestCrypt.shtml), 2005.
- [22] U.S. District Court (Southern District of New York), United States of America v. Robert Johnson (files.findlaw.com/news.findlaw.com/hdocs/docs/chldprn/usjhnsn62805ind.pdf), June 28, 2005.
- [23] U.S. Immigration and Customs Enforcement, U.S. charges ex CEO with using the Internet for child pornography and with obstruction of justice (www.ice.gov/graphics/news/newsreleases/articles/050628newyork.htm), June 28, 2005.
- [24] P. Wayner, *Disappearing Cryptography: Information Hiding, Steganography and Watermarking*, Morgan Kauffman, San Francisco, California, 2002.
- [25] WetStone Technologies, Stego Suite (www.wetstonetech.com).

Chapter 15

ASSESSING TRACE EVIDENCE LEFT BY SECURE DELETION PROGRAMS

Paul Burke and Philip Craiger

Abstract Secure deletion programs purport to permanently erase files from digital media. These programs are used by businesses and individuals to remove sensitive information from media, and by criminals to remove evidence of the tools or fruits of illegal activities. This paper focuses on the trace evidence left by secure deletion programs. In particular, five Windows-based secure deletion programs are tested to determine if they leave identifiable signatures after deleting a file. The results show that the majority of the programs leave identifiable signatures. Moreover, some of the programs do not completely erase file metadata, which enables forensic investigators to extract the name, size, creation date and deletion date of the “deleted” files.

Keywords: Secure deletion, trace evidence, Windows XP, FAT12 file system

1. Introduction

Demand for secure deletion programs has grown with the enormous amounts of sensitive information stored on digital media. Several programs that offer to remove computer users’ digital tracks are being marketed. These programs supposedly delete all traces of files and offer “immunity” from forensic analysis.

This paper evaluates five Windows-based programs based on their ability to delete files from a FAT12 file system. The research goal is to determine what, if any, trace evidence remains after using these programs. Trace evidence is a term used in traditional forensics to refer to evidence left in small, yet measurable, amounts. Our hypothesis is that different programs use different software routines for deleting files. Consequently, the programs may be identified by their signatures left on the file system. From a law enforcement perspective, knowing that a

Please use the following format when citing this chapter:

Burke, P., Craiger, P., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S. (Boston: Springer), pp. 185–195.

suspect used a software program to delete a file at a certain time can be used to establish intent. Therefore, while it may not be possible to easily recover the deleted files, it is useful to extract whatever proof there may be pertaining to the deletion programs that were employed and the files they were used to delete.

The tests are performed on five popular programs:

- Evidence Eliminator 5.0 (Robin Hood Software)
- CyberScrub Privacy Suite 4.0 (CyberScrub)
- East-Tec Eraser 2005 (EAST Technologies)
- UltraSentry 2.0 (IDM Computer Solutions)
- Eraser 5.3 (Sami Tolvanen)

As this paper was nearing completion, we became aware of a study by Geiger and Cranor [2] that examined six secure deletion programs (including two that are evaluated here). Geiger and Cranor investigated the ability of the programs to remove personal data from a Windows-based computer. As such, our work is complementary as it focuses specifically on the residual binary structures, an issue not explored in detail in Geiger and Cranor's work.

2. Background

File systems can be thought of as logical shelves that organize and structure the placement of raw data on digital media. In the grand scheme of information storage they are merely another abstraction layer above the raw data transfers to and from digital media. A file system not only records the location of a file on the media, but also metadata – descriptive information about the file that is not actually part of the file, e.g., filename, timestamps associated with its creation and modification, physical size, and starting location of the file. File systems break up the physical media into logical fixed-size sections called clusters. Each file is allocated a set of clusters in which it is stored. There are several dozen commonly used file systems, each of which structures this information in a different manner.

The family of file systems used with consumer-level Windows versions up until the introduction of Windows XP is called FAT (File Allocation Table) [4]. Several versions of the FAT file system exist, each primarily differing in the number of bits used to address the file system. An area called the root directory stores information pertaining to the metadata of individual files (see Figure 1). Locations of currently utilized clusters are

```

00025f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
0002600: e559 4649 4c45 2020 5458 5420 1814 2056 .YFILE TXT .. V
0002610: e532 e532 0000 d154 e532 0200 9334 0000 .2.2...T.2...4..
0002620: 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

Figure 1. Hex view of the directory entry of a deleted file.

tracked via two identical file allocation tables. A file allocation table is a singly linked list of pointers that identify the specific clusters comprising a file. The actual file contents are stored in the data area elsewhere on the file system.

3. Traditional File Deletion

When a file is deleted in Windows, two changes occur at the file system level. First, both file allocation tables are updated to reflect that the space occupied by the file is now available. Second, the file's directory entry is marked as deleted by overwriting the first character of the filename with 0xE5 as shown in Figure 1. All remaining file metadata in the root directory, including the rest of the filename and its former initial position on the disk, remain unmodified. The contents of the file itself are literally untouched by this process. This is why it is fairly simple to undelete a file – provided that the space that the file occupied on the media has not been overwritten.

The fact that deleting a file does not delete its contents is the basis for the demand of secure deletion software: these programs are designed to permanently remove files. Secure deletion programs go beyond traditional file deletion to prevent file recovery.

4. Secure File Deletion

Most modern media records data in the form of magnetic traces. Thus, a device such as a hard disk can be read at a sufficiently low level to extract the polarity a specific location on the disk had in the past. This is possible because the transition from one polarity to another is not complete – slight magnetic traces remain after a write operation. Thus, specialized equipment can be used to recover data even after it is overwritten.

Several data wiping strategies have been proposed to combat the use of equipment that can read magnetic traces. The general consensus is to perform multiple overwrites using a combination of random characters and zeroes. This data overwriting method has two obvious benefits. First, at some point it becomes infeasible to determine which traces belonged to which write operation. Second, including random

data patterns makes it extremely difficult to determine which traces are the remains of the original data.

Three popular secure deletion strategies related to this general concept are highlighted in the U.S. Department of Defense's National Industrial Security Program Operating Manual (NISPOM) (DoD 5220.22-M) [1] and in a paper by Guttman [3]. Both documents are approximately ten years old, but the techniques they describe are implemented in nearly all modern secure deletion programs.

The 5220.22-M document provides a cleaning/sanitization matrix for different types of media. Within this matrix are two strategies that are often used on magnetic media:

1. Level C: Overwrite all addressable locations with a single character. This is often referred to as "DoD Short." It is usually implemented as a full zeroing of the data area.
2. Level D: Overwrite all addressable locations with a character, its complement, then a random character, and verify that this has occurred. This is often referred to as "DoD 3-pass" (or some variant).

Some secure deletion programs reference a 7-pass DoD secure delete that complies with 5220.22-M, but such a strategy is not detailed in the document. In fact, the February 2001 revision of NISPOM omits all recommendations for low-level sanitization strategies.

Guttman's paper [3] discusses Magnetic Force Microscopy (MFM) that can be used to read deviations in the magnetic force on a hard disk platter and determine data from past writes. Based on the binary encoding that a hard disk drive uses on the platter, Guttman proposes several permutations of binary patterns (35 in all, eight of which involve random data) for the purpose of reliably erasing data from various hard disks. A 35-pass technique implementing all of Guttman's suggestions is commonly used by secure deletion programs. However, as Guttman notes in a follow-up to his original paper [3], it is only necessary to use a subset of his suggestions for the disk drives in use today. In fact, due to the encoding schemes used in drives manufactured within the past ten years, multiple overwrites with random data is the most effective secure deletion strategy.

5. Complete Deletion Protocol

Based on our research, we posit that a secure deletion program must perform four operations on a file in order to assure its complete removal from a FAT file system.

1. Identify the physical locations of the file on the media and overwrite those areas to the end of the occupied clusters, performing either a single pass or multiple passes with all 0s, all 1s, a random data pattern, or a combination thereof.
2. Locate the file's root directory entry and remove all metadata associated with the file by overwriting with random data in multiple passes. Data in the entry should be carefully maintained to appear correlated with a regular file delete, e.g., date modified should not be after the current date.
3. Locate and remove the file allocation table entries for the file using multiple passes, freeing up the space used by the file at the logical level.
4. Purge all the information regarding the file that remains in memory. Depending on the operating system in use, this may require the computer to be rebooted.

6. Testing Procedure

A series of tests were performed to determine what trace evidence remained on a file system after a secure deletion operation was performed on a file. All testing occurred on a computer running Windows XP (Service Pack 2). The following protocol was used to develop the test media:

1. Overwrite a 3.5" floppy disk with a pattern of binary zeroes using the GNU `dd` program in Linux.
2. Format the floppy disk with a full format in Windows XP to create a FAT12 file system on the floppy disk.
3. Copy a single text file (`myfile.txt`) of size 13459 bytes to the floppy disk.
4. Eject and image the floppy on a separate computer.

Five identical tests were performed with each program to determine what, if anything, varied after the secure deletion process. By comparing each individual test it was possible to determine which elements of the deletion process were constant and which were random. The following protocol was used for each run:

1. Insert the floppy disk into the disk drive and start the secure deletion program.

2. Select the file to be securely deleted on the floppy disk and run the secure deletion operation on it.
3. Image the floppy using the GNU `dd` program in Linux.
4. Wipe the floppy with a pattern of binary zeroes to remove any remaining trace evidence, and load the original image onto the floppy using the GNU `dd` program in Linux.
5. Reboot the computer and repeat the first step for the next test.

The five acquired images were then compared to analyze the variations in the results produced by each secure deletion program.

A separate Windows XP install was created for each program to eliminate potential conflicts between programs. KNOPPIX 3.8.1 was used to image the partition for each install. All systems involved in data acquisition were isolated from the Internet to prevent potential contamination. Analyses were performed at the binary level using the console hex editors `xxd` and `htex`; `cmp` (part of the GNU `diff` suite) was used to determine any changes between tests. Additionally, Sleuth Kit and Forensic Toolkit (FTK) were used to confirm the initial findings at a higher level.

The default preferences for all programs were retained. No modifications were made to the program preferences after the initial installation process. Due to the diversity of options and functions available in the programs, it was not feasible to try to match up similar erasing techniques. The assumption is that most users do not modify the default preferences, particularly as one must have some technical knowledge to understand the differences between deletion strategies.

Table 1. Default secure deletion protocols.

Secure Deletion Program	Default Protocol
Evidence Eliminator	Zeroes file (1 pass)
UltraSentry	DoD Level C
CyberScrub	DoD 5220.22-M (7 passes)
Eraser	Guttman (35 passes)

As the analysis was not performed at the raw magnetic level, no statements can be made here about the effectiveness of each program in that regard. All the programs deleted the file with at least one pass. Table 1 summarizes the default protocols used by the secure deletion programs.

7. Results

As expected, all the programs completely deleted the file contents. However, several programs did leave digital signatures within the file's root directory entry. The findings are presented below.

7.1 Evidence Eliminator

Robin Hood Software's Evidence Eliminator 5.0 offers numerous anti-forensic features, including several different methods for erasing trace evidence left during regular Windows use. In addition, it provides a single-file secure deletion operation, the function tested in this study. The test was conducted under "Quick Mode" because it does not require a restart to complete the erasing process and is the option most likely to be chosen by a novice user.

```

00025f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
0002600: e559 4649 4c45 2020 5458 5420 1814 2056 .YFILE TXT .. V
0002610: e532 e532 0000 4e5b e532 0200 0080 0000 .2.2..N[.2.....
0002620: e545 0045 002d 002d 000f 00aa 2d00 .E.E.-.-.-.-.-.
0002630: 2d00 2d00 2d00 2e00 7400 0000 6d00 7000 -.-.-...t...m.p.
0002640: e545 2d2d 2d2d 7e31 544d 5020 0014 2056 .E-----1TMP .. V
0002650: e532 e532 0000 4f5b e532 0200 0100 0000 .2.2..0[.2.....
0002660: 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

Figure 2. Residual evidence left by Evidence Eliminator.

Evidence Eliminator left a conspicuous signature: it created an additional temporary file on the file system, something no other program appears to have done (see Figure 2). This file is consistently named `EE-----1.TMP` (short filename: `?E-----1.TMP`, first character deleted) as reported by Sleuth Kit's `fls`, which lists filenames in a file system image. Windows is apparently left in control of the attributes for both files so the date modified attribute of the files indicates the approximate time the file was deleted. The temporary file has the same creation date as the deleted file.

The file itself is marked as deleted in the traditional manner, using `0xE5` as an indicator of its status. Evidence Eliminator leaves the rest of the filename and its attributes untouched. This is oddly inconsistent with the program's default preferences, which state "For extra security, rename and zero sizes when wiping files." The only piece of information in the directory entry that distinguishes an Evidence Eliminator deletion from a standard Recycle Bin deletion is the file size, which was modified in the test case to a full 32 KB (from its original 13.1 KB). The file's creation date is preserved and, as noted above, the written

Table 2. Evidence Eliminator summary.

Filename	?yfile.txt
Creation Date	Unmodified
Modification Date	Date of file deletion
File Attributes	Unmodified
Logical Size	Increased to 32768 Bytes
Data Area	Zeroed out
Notes	Added temporary file named EE-----.tmp (short: ?E----~1.TMP) with the same creation date and similar written times

attribute indicates the time when the file was deleted. The written date of the temporary file listed above and the file itself are usually only seconds apart, most likely depending on the speed of the delete operation. The data area of the file was zeroed out. The test results for Evidence Eliminator are summarized in Table 2.

7.2 UltraSentry

IDM Computer Solutions' UltraSentry 2.0 offers the standard array of features for purging trace evidence. In addition to offering the usual set of deletion strategies, UltraSentry permits a user to customize file deletion by specifying the number and types of passes on a file.

```

00025f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
0002600: e550 5050 5050 2020 5050 5020 1814 2056 .PPPPP PPP .. V
0002610: e532 e832 0000 755a e832 0200 9334 0000 .2.2..uZ.2...4..
0002620: 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

Figure 3. Residual evidence left by UltraSentry.

UltraSentry makes an attempt at hiding the filename by overwriting each character in the filename with an upper case P (0x50), in addition to marking it as deleted (first character: 0xE5) (see Figure 3).

The number of characters used in the new filename appears to remain consistent with the original. The file written time is modified to reflect the time that the file was deleted; the creation time and file size are unchanged. Like Evidence Eliminator, UltraSentry zeroes out the data area where the file existed. The test results for UltraSentry are summarized in Table 3.

Table 3. UltraSentry summary.

Filename	?PPPPP.PPP
Creation Date	Unmodified
Modification Date	Date of file deletion
File Attributes	Unmodified
Logical Size	Unmodified
Data Area	Zeroed out
Notes	Number of letters used to overwrite the filename is the same as the length of the original filename

7.3 CyberScrub and East-Tec Eraser

CyberScrub's Privacy Suite 4.0 and EAST Technologies' East-Tec Eraser 2005 are grouped together because they are functionally similar. In fact, the two programs are almost identical down to the online registration system. Some of the file sizes for the programs and their associated libraries differ, but the two programs are the same in terms of their user interfaces and end results.

```

00025f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
0002600: e537 3037 3732 3539 3346 3222 00c6 0180 .70772593F2"....
0002610: af2a af2a 0000 0280 af2a 0000 0000 0000 *.*.....*.....
0002620: e538 3736 3331 3732 5749 5022 10ab e172 .8763172WIP"...r
0002630: e732 e732 0000 5973 e732 0000 0000 0000 .2.2..Ys.2.....
0002640: e538 3132 3746 4639 3346 3222 00c7 af5e .8127FF93F2"...^
0002650: 9a2e 9a2e 0000 b05e 9a2e 0000 0000 0000 .....^.....
0002660: e538 3439 3337 3444 3346 3222 00b5 c572 .849374D3F2"...r
0002670: eb2c eb2c 0000 c672 eb2c 0000 0000 0000 ,.,.,.,.r.,.....

```

Figure 4. Directory entries resulting from a CyberScrub deletion.

The CyberScrub and East-Tec Eraser programs go far as to overwrite all unused entries in the directory entry area with what appears to be random data. Each deleted file's root directory entry is overwritten with a deleted file with a random filename with the same three-letter extension .WIP (see Figure 4).

The date and timestamp of each file appears to be randomly selected from dates prior to the date of deletion, and the size of the file is set to zero. Unlike the other secure deletion programs, CyberScrub and East-Tec Eraser set the deleted file's Hidden attribute. Similar to the other programs, the file's data area is filled with random data up to the end of the file's last cluster. The test results for CyberScrub and East-Tec Eraser are summarized in Table 4.

Table 4. CyberScrub/East-Tec Eraser summary.

Filename	Name and extension of the deleted file are overwritten with random characters
Creation Date	Randomized, but never after current date
Modification Date	Within 2 seconds of creation date
File Attributes	Added hidden attribute
Logical Size	Increased to 32768 Bytes
Data Area	Randomized to the end of the file's last cluster
Notes	Directory entries are filled with deleted entries, each having the same three-character extension; one entry has the .wip extension

7.4 Eraser

Sami Tolvanen's Eraser 5.3 is provided under the GNU General Public License and is available free-of-charge to the public. The version tested was the final version released by Tolvanen before maintainership was transferred to Garrett Trant of Heidi Computers Limited. Several versions of the program have since been released, the most recent being Eraser 5.7. Despite its age, the Eraser 5.3 program is compatible with the latest version of Windows.

```

00025f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
0002600: e541 315a 544f 2020 554c 4520 0000 0000 .A1ZT0 ULE ....
0002610: 2100 2100 0000 0000 2100 0000 0000 0000 !.!.!.!.!.!.!.
0002620: 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

Figure 5. Residual evidence left by Eraser.

Eraser deleted the test file, randomizing its filename but keeping the number of characters used for the filename the same as that of the original file (see Figure 5). The file extension is similarly randomized. The date attributes of the file, instead of being zeroed out or set to a random date, are consistently set to midnight, January 1, 1980 (corresponding to the start of the FAT file system epoch). The file size is set to zero, and the data area is filled with random bits up to the end of the cluster in which the file existed. The test results for Eraser are summarized in Table 5.

8. Conclusions

Secure deletion programs claim to permanently remove files from digital media. However, tests of five popular Windows-based programs demonstrate that each program leaves unique signatures, which could

Table 5. Eraser summary.

Filename	Name and extension of the deleted file are overwritten with random characters
Creation Date	January 1, 1980
Modification Date	January 1, 1980
File Attributes	Unmodified
Logical Size	Null
Data Area	Randomized to the end of the file's last cluster
Notes	Date fields are set to the start of the FAT file system epoch rather than being zeroed out

assist examiners in determining whether a secure deletion was performed and in identifying the program used to perform the deletion.

Our tests did not investigate the numerous options provided by the secure deletion programs. In fact, the testing used the default preferences, which are not necessarily optimized for effectiveness. Several of the program options, if properly utilized, may alter or remove the trace evidence found during testing. We propose to extend this research to other file systems (e.g., FAT32, NTFS, ext2), other operating systems (Windows, Linux, UNIX, Mac OS X), as well as the full cadre of secure deletion utilities that exist for these operating systems.

References

- [1] Defense Security Service, *National Industrial Security Program Operating Manual (NISPOM)*, DoD 5220.22-M, U.S. Department of Defense (www.dss.mil/isec/nispom_0195.pdf), 1995.
- [2] M. Geiger and L. Cranor, Counter-Forensic Privacy Tools: A Forensic Evaluation, Technical Report CMU-ISRI-05-119, Institute for Software Research International, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania (reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-119.pdf), 2005.
- [3] P. Guttman, Secure deletion of data from magnetic and solid-state memory, *Proceedings of the Sixth USENIX Security Symposium* (www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html), 1996.
- [4] Microsoft Corporation, FAT32 File System Specification (www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx), 2000.

Chapter 16

ON THE RELIABILITY OF NETWORK EAVESDROPPING TOOLS

Eric Cronin, Micah Sherr and Matthew Blaze

Abstract This paper analyzes the problem of intercepting Internet traffic from the eavesdropper's point of view. It examines the reliability and accuracy of transcripts, and shows that obtaining "high fidelity" transcripts is harder than previously assumed. Even in highly favorable situations, such as capturing unencrypted traffic using standard protocols, simple – and entirely unilateral – countermeasures are shown to be sufficient to prevent accurate traffic analysis in many Internet interception configurations. In particular, these countermeasures were successful against every available eavesdropping system we tested. Central to our approach is a new class of "confusion" techniques, that unlike cryptography or steganography, do not require cooperation by the communicating parties and, in some cases, can be employed entirely by a third party who is not involved in the communication.

Keywords: Eavesdropping, electronic interception, countermeasures

1. Introduction

The results of Internet interceptions are almost always accepted uncritically. While previous work has shown the potential for spurious errors [1, 3] or evasion [13, 15] to interfere with capture, there has been remarkably little exploration of the problems that face eavesdroppers who wish to ensure the accuracy of their intercepts. We assert that the eavesdropping task is far more difficult than has previously been realized, and show that existing tools are insufficient to gauge the accuracy of captured traffic.

At least six properties of the Internet protocol stack and architecture make it difficult for an eavesdropper to accurately reconstruct communications from intercepts. They include decentralized control and heterogeneous implementations; "best effort" (as opposed to reliable) message

Please use the following format when citing this chapter:

Cronin, E., Sherr, M., Blaze, M., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 199–213.

delivery that allows data to be re-ordered, duplicated or dropped in transit; shared state and context between communicating parties; dynamic (and often asymmetric) routing that can change during a flow's lifetime; lack of sender and receiver authentication; and ambiguities in protocols, implementations and configurations.

These properties mean that a great deal of state information is involved in the correct interpretation of any given packet, and this state is spread across many places, including each of the communicating parties and the network itself. Without complete knowledge of this state, the mere presence of a packet somewhere on the network does not automatically imply that it will be accepted by the recipient given in its header, that it came from the supposed sender, or that it has not been (or will not be) altered, duplicated or deleted somewhere along its path.

Any intercept system must take into account these properties (and all the corresponding states) to ensure not only that it is sufficiently "sensitive" (that it receives all data exchanged between the targets), but that it is also sufficiently "selective" (that it rejects spurious data that is not actually part of the targets' exchange) [4]. The figure of merit most often considered in judging intercept systems is sensitivity. Adequate selectivity, on the other hand, is generally thought to be easily achieved by cursory examination of, for example, packet headers. In fact, selectivity may be a far more difficult problem than most intercept systems recognize, especially in the presence of deliberate countermeasures.

Fortunately for the eavesdropper, on more benign networks, many of the factors that might introduce uncertainty about the veracity and interpretation of a given packet are relatively static, at least for the lifetime of a particular interception. For example, although routes can theoretically change midstream, in practice, they rarely do; and although routers and hosts are free to alter, reorder, delay and duplicate packets, for the most part they refrain from doing so.

However, this lends a false sense of security to those producing eavesdropping tools. Depending on the network configuration, many ambiguities can be intentionally induced, either by one of the communicating parties or by a third party. In fact, across much of the protocol stack, from the physical layer to applications, it is surprisingly simple to introduce data that appears entirely valid but that might not be received and processed by the purported recipient. The Internet appears almost to have been designed to maximize uncertainty from the point of view of those eavesdropping on it.

In particular, we observe that a single party, which we call a "confuser," can introduce traffic directed at an eavesdropper but that is never actually received (or if received, is rejected) by the ostensible recipient.

Depending on the eavesdropper's configuration and position in the network, this traffic can be made indistinguishable from legitimate traffic. In the presence of sufficient confusion, an eavesdropper could be made arbitrarily uncertain as to whether a given intercepted message was real or spurious.

We introduce some terminology that will be used throughout this paper. As is customary, Alice and Bob will represent our network communicators. Alice will often be a source while Bob will be a sink (in most protocols the roles are symmetric and often alternating). Eve will be the eavesdropper. An interception system is vulnerable to confusion if it captures and records in its transcripts messages that are purportedly from Alice to Bob but that are rejected or otherwise not processed by Bob.

Although we do not advocate that confusion be used as a general confidentiality technique, we note that confusion has some interesting qualities that make it attractive as an eavesdropping countermeasure.

- While cryptography is typically used to ensure the confidentiality of message payloads, confusion protects both message contents and metadata. It may, therefore, be advantageous to combine confusion with encryption to mask the signaling information as well as the content.
- Since confusion is transparent to Bob, it may be easily incorporated into existing protocols. Thus, it may be particularly useful when legacy applications and protocols cannot be easily upgraded or replaced.
- If the confuser is a third party, then neither Alice nor Bob need to be aware of the confusion. Unlike bilateral techniques in which it is obvious that Alice and Bob have colluded to disguise their messages, confusion allows Alice and Bob to deny that they even attempted to communicate privately.

2. Related Work

There has been little prior work investigating the general problem of traffic interception from the eavesdropper's point of view [1, 3, 5]. However, considerable research has addressed the related topic of information privacy. Cryptography, steganography, subliminal or covert channels [20], winnowing and chaffing [17], quantum communication [2] and anonymous communications [7, 16], for example, all focus on establishing confidential communication.

Work from the eavesdropper's point of view has primarily been limited to the specialized area of intrusion detection [12, 13, 18]. In a network intrusion detection system (NIDS), the primary goal of the listener (eavesdropper) is real-time analysis of incoming traffic to recognize attack signatures and detect anomalies. These systems are deployed at the borders of controlled networks where it becomes much easier to make assumptions about the machines within the network that the system protects. Additionally, the communication patterns of an attacker are also unique compared to general bidirectional communications; hence the NIDS can flag suspicious traffic. However, unlike a NIDS, a general purpose eavesdropper must process all traffic, both normal and anomalous. It is possible to draw some results from NIDS research, but the applicability is limited by the different constraints on topology and communication characteristics.

3. Confusion in the Internet Architecture

The Internet is, by design, a very heterogeneous system. Machines of differing hardware and software configurations communicate and interoperate through the use of standard protocols. However, ambiguities in implementations, configurations and protocol specifications create the opportunity for non-uniformity in the processing of specially-crafted messages. Confusion exploits these inconsistencies by forcing an eavesdropper to consider multiple plausible interpretations of its transcripts. The IP and TCP specifications (which famously advise "be conservative in what you do, be liberal in what you accept from others" [14]) aggravate the problem of proper selectivity by recommending that implementations accept even outlier communications.

Below, we explore various vectors and techniques for injecting confusion in the Internet architecture. The confusion countermeasures are not intended to be exhaustive; rather, their purpose is to illustrate the ease and effectiveness with which reliable interception can be defeated.

3.1 Physical Layer Confusion

At the physical layer, network devices convert analog signals into digital encodings. To allow interoperable devices, standards exist that define acceptable ranges for amplitudes, frequencies, voltages, etc. [8–10]. However, because transmission and decoding are analog processes, for any given parameter (frequency, amplitude, etc.), no two decoders will use the same threshold to determine whether a given signal is accepted or rejected. Thus, network devices, especially commodity hardware, do

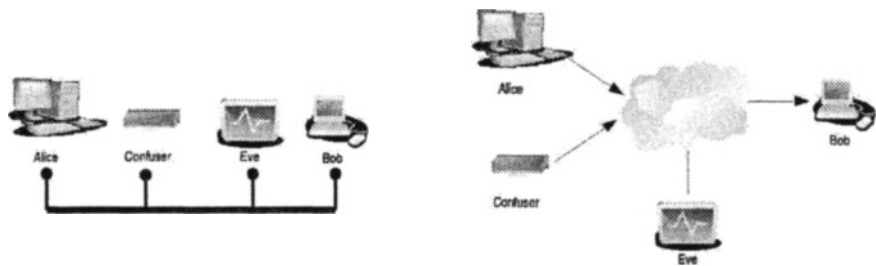


Figure 1. Left: All parties communicate using a shared bus; Right: Eve is located between Alice and Bob.

not strictly abide by these standards and often interpret messages sent outside of the specified ranges.

Alice can exploit these differences to evade as well as confuse Eve. As depicted in the left-hand side of Figure 1, we assume a topology in which all parties share the same communication medium (e.g., a common bus or a wireless network). To evade Eve, Alice can transmit messages at a frequency, amplitude or voltage that are imperceptible to Eve but acceptable by Bob. Note that this type of physical evasion is more difficult when Alice, Bob and Eve do not share a communication medium, as intermediary routers act as normalizers and reduce the likelihood of an effective evasion attack. Generally, if Eve is less sensitive than Bob and the three parties share a communication medium, then Eve is susceptible to evasion.

Eve's obvious counter-countermeasure (i.e., enhancing her sensitivity) has the unfortunate effect of increasing her vulnerability to confusion [4]. If Eve is more sensitive than Bob, evasion is not possible. However, a third-party confuser can now inject noise that is processed by Eve but ignored by Bob. As a result, Eve is forced to consider multiple interpretations, while Bob only sees the legitimate messages.

3.2 Link Layer Confusion

Confusion is possible at the link layer if the confuser and Eve share the same Ethernet. A typical example of such a topology is an unencrypted 802.11 network in which Eve “sniffs” wireless transmissions.

We show empirically in Section 4 that current eavesdropping systems suffer from inadequate selectivity. Although most eavesdropping systems can record traffic at the link layer, they often ignore Ethernet frames and instead process messages at the network or transport layer. By crafting Ethernet frames with invalid MAC destination addresses, a confuser can inject noise that is processed by Eve but is not delivered to Bob [15].

Neither Bob nor the local gateway process the noise as their operating systems silently discard Ethernet frames whose MAC addresses do not match those of the network interface.

This technique is only effective when Eve has poor selectivity. If Eve examined the Ethernet frames, she would be capable of distinguishing the noise from the message text. Unlike other confusion countermeasures, the MAC technique is not indicative of a fundamental limitation of electronic eavesdropping. However, the significance of the approach is that it illustrates the dangers of inadequate selectivity: an eavesdropping system that fails to properly process Ethernet frames is inherently vulnerable to this form of confusion. Accordingly, an Internet eavesdropping system that observes traffic on a local Ethernet cannot claim to be reliable unless it intercepts and processes link layer headers.

3.3 Network Layer Confusion

If Eve intercepts a packet on the path from Alice and Bob (right-hand side of Figure 1), she must carefully examine the packet's IP header to form an opinion as to whether the packet is deliverable. A packet may not be delivered for several reasons: the packet checksum may be incorrect, IP options may be specified that are unsupported by an intermediary router (e.g., source routing), the packet size may exceed the MTU of a hop, or the initial time-to-live (TTL) value may be insufficient to reach Bob [14, 15]. If the confuser has more knowledge about the network than Eve, he can inject noise that will be dropped either before reaching Bob or by Bob's IP implementation. If Eve processes all intercepted IP packets, which – as we show in Section 4 – is the case with all tested eavesdropping systems, then she will interpret the noise along with the legitimate traffic.

As with link layer techniques, network layer confusion countermeasures highlight weaknesses in current eavesdropping systems. By enhancing Eve's selectivity, many of these countermeasures can be eliminated. However, an eavesdropper, who does not examine IP headers or lacks enough selectivity to determine whether packets are deliverable, is inherently vulnerable to this type of confusion.

4. Failure of Current Eavesdropping Systems

In this section we examine common eavesdropping tools in several domains, and show how they are vulnerable to simple, unilateral attacks. We look at examples of digital and analog tools.

Open Source Eavesdropping Tools

- **Bro:** Bro is a network intrusion detection system developed at the University of California, Berkeley. As such, it does not operate as an eavesdropping tool by default. However, it has a very robust stream reconstruction engine, and can be cajoled into acting as an offline analysis tool. We ran Bro using the 'weird,' 'conn,' 'contents,' 'frag' and 'smtp' policies with their default settings. Bro can be found at: www.bro-ids.org.
- **Chaosreader:** Chaosreader is a user-friendly TCP reconstruction tool that creates HTML pages for the contents of intercepted sessions. It can be found at: chaosreader.sourceforge.net.
- **Ethereal:** Ethereal is a popular eavesdropping tool. Although most of its features are packet oriented, it contains a TCP reassembly option that was used in the experiments. Ethereal can be found at: www.ethereal.com.
- **Snort:** Snort is a commonly used NIDS. We ran it in offline mode using the stream4 and stream4_reassemble preprocessors with the log_flushed_streams option. In addition, we used the snort-replay patch, which uses its own stream reconstruction implementation. Snort can be found at: www.snort.org, and snort-replay: at www.algonet.se/~nitzer/snort-replay.
- **tcpick:** tcpick is a pcap-based packet sniffer and TCP reconstruction tool. It can be found at: tcpick.sourceforge.net.
- **tcptrace:** tcptrace is an analysis tool for pcap-based network intercepts. Among its many features, tcptrace can reconstruct captured TCP streams. It can be found at: jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html.
- **tcpflow:** tcpflow is a useful tool for conducting TCP stream reassembly. It operates by processing pcap dump files and extracting the contents of TCP streams. It can be found at: www.circlemud.org/~jelson/software/tcpflow.

Commercial Eavesdropping Tools

- **CommView:** CommView is a commercial Windows eavesdropping tool. An evaluation version can be found at: www.tamos.com/products/commview.
- **NetworkActiv PIAFCTM:** PIAFCTM is a commercial Windows eavesdropping tool. A trial version is available at: www.networkactiv.com/PIAFCTM.html.
- **Sniffem:** Sniffem is a commercial Windows eavesdropping tool. A trial version is available at: www.sniff-em.com/sniffem.shtml.

Figure 2. Eavesdropping tools evaluated in this work.

4.1 Digital Eavesdropping Evasion

To demonstrate the susceptibility of current eavesdropping tools (Figure 2) to confusion, we implemented the MAC and TTL confusion techniques described in Section 3 and originally introduced as NIDS attacks in [15]. Note that fragroute [21] also provides an implementation of the NIDS techniques, but it was found to be unsuitable for general purpose bidirectional communication. The MAC approach relies on generating noise with invalid MAC destination addresses. While Eve will process the noise, the local gateway will not route such packets since it only accepts correctly addressed Ethernet frames. In the TTL technique, the

confuser introduces noise with TTLs that are sufficient to reach Eve but not Bob. Note that both techniques can be trivially defeated by providing adequate selectivity. Here, our aim is not to introduce formidable countermeasures. Rather, we show that current eavesdropping tools are susceptible to even weak forms of confusion.

In our experiments, Alice transmits an email via SMTP to our institution's email server (Bob). To confuse Eve, Alice (who functions as the confuser) injects spurious noise using either the MAC or the TTL confusion techniques. To maximize confusion, Alice sends the legitimate email and the noise in byte-sized packets (since TCP is stream based, applications that rely on TCP are generally unaffected by the size of the transmitted packets). For every byte of legitimate text, Alice sends eight noise packets. Of the eight noise streams, the first comprises a "cover message." This first stream, although composed of noise, constitutes a false but sensible message – a passage from Dickens' *A Tale of Two Cities* [6]. The remaining seven streams of noise consist of random characters. In an attempt to cause Eve to interpret the false stream rather than her true message, Alice always sends the false stream first, followed by the random intermixing of the legitimate stream and the seven random noise streams. No modifications were made to the SMTP server (Bob).

We tested our link and network layer confusion tools against 11 eavesdropping systems, ranging from commercial applications to free open-source toolkits (Figure 2). Experiments were conducted on a network test bed in which Alice and Eve reside on the same local subnet. From this subnet, a minimum TTL of five is required to reach Bob. Both Alice and Eve are Pentium servers with 3COM Fast EtherLink XL 100MB/s network cards that are connected via a 100MB/s switch.

The performance of the eavesdroppers in the presence of confusion was startlingly lacking. Table 1 describes Eve's (in)ability to reliably reconstruct email messages. Although all but one eavesdropping package could reconstruct Alice's message in the absence of confusion, all the tested systems failed to interpret her message when either of the two confusion techniques was applied. Anomalies were reported by only 18% of the eavesdroppers with the MAC-based approach and 27% of the systems when TTL confusion was used. Moreover, the cover message was perceived as the email in 45% of the cases when either technique was utilized (Figure 3). In all cases, the email server (Bob) correctly received Alice's communication and delivered the email to its intended recipient.

Table 1. Ineffectiveness of various eavesdropping tools against confusion techniques. ✓: Tool correctly interpreted the message; $\times_C$: Tool incorrectly interpreted cover message as legitimate message (see Figure 3); $\times_R$: No discernible English text obtained from the eavesdropper's interpretation; *DUP*: TCP DUPs detected; *TTL*: TTL exceeded; *RI*: Retransmission inconsistency.

Software	No Confusion		MAC Confusion		TTL Confusion	
	Valid	Errors	Valid	Errors	Valid	Errors
Bro	✓	—	$\times_C$	<i>RI</i>	$\times_C$	<i>RI</i>
chaosreader	✓	—	$\times_R$	—	$\times_R$	—
CommView	✓	—	$\times_C$	—	$\times_C$	—
Ethereal	✓	—	$\times_C$	—	$\times_C$	—
PIAFCTM	✓	—	$\times_C$	—	$\times_C$	—
Sniffem	$\times_R$	—	$\times_R$	—	$\times_R$	—
snort-replay	✓	—	$\times_R$	—	$\times_R$	—
snort-stream4	✓	—	$\times_R$	—	$\times_R$	<i>TTL</i>
tcpick	✓	—	$\times_C$	—	$\times_C$	—
tcptrace	✓	—	$\times_R$	<i>DUP</i>	$\times_R$	<i>DUP</i>
tcpflow	✓	—	$\times_R$	—	$\times_R$	—

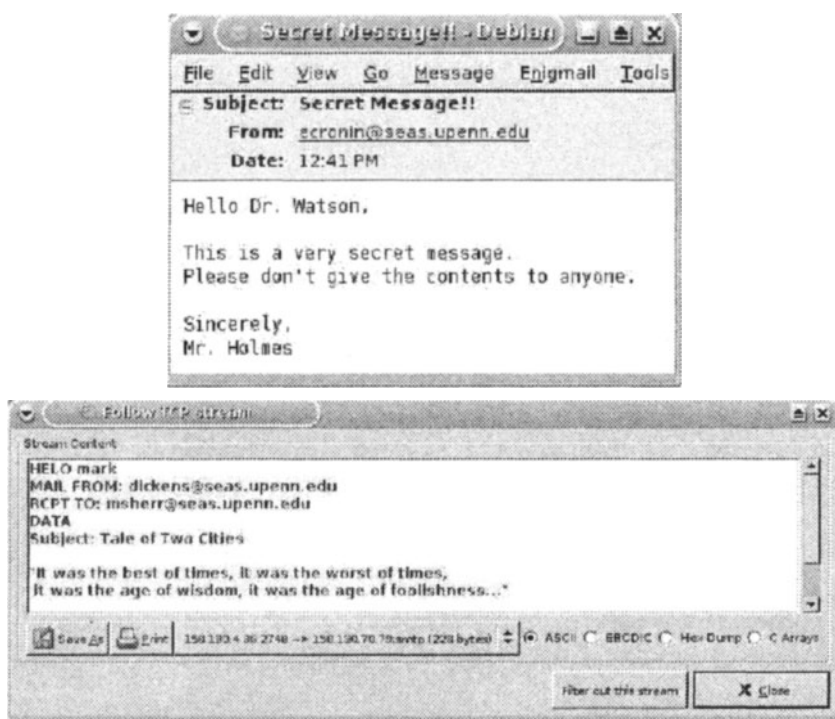


Figure 3. Top: Legitimate message received by the SMTP server (Bob) and the intended email recipient; Bottom: Ethereal reconstruction in which Eve fails to capture Alice's message and perceives the cover message as the legitimate message.

4.2 POTS Evasion and Confusion

Confusion and evasion can be practical threats to digital Internet eavesdropping, and indeed, such systems are the focus of this paper. However, similar techniques can also be applied to analog networks, especially when analog to digital conversion is performed. As an example, we consider voice telephone signaling between the subscriber and the switch.

Analog telephone service, also known as “plain old telephone service” (POTS), uses analog touch-tones to signal the caller’s desired number. The touch-tone system is based on the international DTMF standard [11]. Each of the 16 DTMF digits is composed of two base frequencies. The specification lists acceptable ranges for tone duration, spacing, frequency, amplitude and twist (relative amplitude of the high and low frequencies).

Constructing a decoder for DTMF signals that satisfies the specifications exactly is surprisingly difficult and expensive. Instead, most decoders relax the tolerances in one or more of the dimensions of the standard, accepting tones that are a little too loud or quiet, or a little out of pitch, etc. Since each decoder has a unique range of acceptable tones, it is possible to construct out-of-specification tones that some decoders will see but others will miss. Using this knowledge, it is possible to evade and confuse eavesdroppers.

To test the practicality of this countermeasure, we constructed a simulated telephone network. A Teltone TLS-5 Telephone Line Simulator was used as the central office, providing dial tone and four phone lines with unique phone numbers, as well as DTMF switching to connect calls between lines. Alice, the call initiator, was an Ameritec AM8a PCM/VF call analyzer with an ordinary POTS handset connected to it; Bob was a second POTS handset; and Eve was a Metrotel VNA 70a DTMF decoder.

Among its many features, the AM8a call analyzer used by Alice allows DTMF codes to be generated with precise, configurable parameters. In our experiments, we focused on two variables: the frequency and amplitude of the higher of the two DTMF frequencies. Settings that allowed for both evasion and confusion were discovered using binary search. Evasion can be applied by decreasing the amplitude of the higher frequency. At -39 dBm, the central office correctly decoded Alice’s signal and completed the call, while the eavesdropper recorded nothing. Similarly, when the high frequency was increased by 3 Hz, the central office no longer recognized Alice’s touch-tones, but the eavesdropper recorded them as having been dialed. Using Alice’s handset in

coordination with the AM8a, the legitimate number could be dialed and interspersed with out-of-range digits to provide confusion. In addition, although we did not test the scenario, by combining both techniques it is clear that Alice could drive Eve to a specific false phone number.

This experiment highlights the challenges faced by an eavesdropper positioned too close to the sender. Limited sensitivity and imperfect selectivity make it susceptible to evasion and confusion countermeasures. While Eve may be certain that the intercepts originated from Alice, she cannot be certain where they terminate in the telephone network. A far more reliable form of dialed number recording is, therefore, achieved through analysis of call detail records generated by the switch itself, but this is, of course, not surreptitious with respect to the operators of the switch. See [19] for details about using confusion as a telephone wiretapping countermeasure.

5. Improving Eavesdropping Reliability

The experiments described in the previous section show how unilateral countermeasures can reduce the reliability of eavesdropping systems. This section explores methods to improve the resilience of eavesdropping tools to such countermeasures.

5.1 Enhancing Sensitivity

To reduce her susceptibility to evasion, Eve can improve her sensitivity. This implies recording at the lowest possible OSI layer, and recording everything that is available (even data that appears to be erroneous). Any action that could have been performed automatically by the lower layers, such as discarding corrupt packets, can be carefully emulated by Eve in a more selective manner.

Unfortunately, this approach may be hard to implement. For example, many authorized uses of eavesdropping in the United States operate under strict limitations on what can be recorded to prevent the traffic of those who are not under suspicion from being observed. In such an environment, the steps that Eve can take to improve her sensitivity are reduced.

5.2 Enhancing Confusion Detection and Eavesdropper Selectivity

In some situations, confusion may be made ineffective by deploying confusion-aware eavesdroppers. For example, the MAC confusion technique described in Section 3 can be defeated with improved software. By

enhancing her sensitivity, Eve may be able to better identify and filter the noise, thereby improving her reliability. However, if Eve is careless in her selections and ignores packets with covert information, she provides Alice and Bob with an unmonitored communication channel.

5.3 Active Eavesdropping

Confusion is only possible when there is an asymmetry in knowledge between Eve and the confuser. To inject uncertainty in Eve's transcripts, the confuser exploits his knowledge (e.g., the network topology or Bob's TCP/IP stack configuration) to ensure that the noise is removed or filtered before being processed by Bob. If Eve can also acquire this knowledge, she can apply the same filter and can therefore trivially defeat confusion.

The intuitive solution to constructing a confusion-resistant eavesdropper is to make Eve active. In addition to passively observing traffic, an active eavesdropper attempts to learn more about the network and the communicating parties by sending out probes. For example, an active eavesdropper can counter the TTL confusion technique described in Section 3 by counting the number of network hops to Bob. By acquiring additional knowledge, Eve can improve her selectivity and overall reliability.

Unfortunately, active eavesdropping does not always ensure the reliable reconstruction of intercepted traffic. First, the probes used by an active Eve can themselves be subjected to a form of confusion. As a counter-counter-countermeasure, a confuser can inject a number of fake responses to Eve's probes. Returning to the TTL confusion example, a confuser can transmit fake ICMP TTL-exceeded messages to frustrate Eve's ability to discern the true TTL cutoff. Second, if Eve actively transmits probes, she may reveal her presence to Alice, Bob and/or the confuser. Since eavesdropping is usually meant to be clandestine, active eavesdropping may be inappropriate in many situations.

5.4 Improving Reliability via Eavesdropper Placement

The location of Eve in the network topology may affect her resilience to confusion. An intuitive approach is to position her in close proximity to Alice. The ability of distant third-party confusers to inject noise is thus diminished as Eve can better discern Alice's communications from those of a distant forger. Unfortunately, this strategy is ineffective when Alice functions as the confuser. Unless Eve can determine which of

Alice's messages are authentic, her position does little to improve her reliability.

A better solution is to place Eve as close as possible to Bob (and as far as possible from any confusers). For example, the TTL confusion technique is ineffective if Bob and Eve reside on the same local network. A disadvantage of this approach is that Eve can only make reliable claims about the messages received by Bob. Her distance from Alice may make the authenticity of intercepted messages harder to establish.

A more ideal strategy is to deploy a number of collaborating eavesdroppers throughout the network. By comparing messages intercepted near the sender versus the receiver, Eve may be able to remove likely noise and improve her reliability. The analysis of colluding eavesdropping is an area of future research.

6. Conclusions

For electronic wiretapping systems to be reliable, they must exhibit correct behavior with regard to both sensitivity and selectivity. Since capturing traffic is a requisite of any monitoring system, considerable research has focused on preventing evasion attacks and otherwise improving sensitivity. However, little attention has been paid to enhancing selectivity or even recognizing the issue in the Internet context.

Traditional wisdom has held that eavesdropping is sufficiently reliable as long as the communicating parties do not participate in bilateral efforts to conceal their messages. We have demonstrated that even in the absence of cooperation between the communicating endpoints, reliable Internet eavesdropping is more difficult than simply capturing packets. If an eavesdropper cannot definitively and correctly select the pertinent messages from the captured traffic, the validity of the reconstructed conversation can be called into question. By injecting noise into the communication channel, unilateral or third-party confusion can make the selectivity process much more difficult, diminishing the reliability of electronic eavesdropping.

Whether eavesdropping can be performed reliably and confusion detected correctly and rejected on the Internet depend heavily on the specific interception topology and on the locations of the potential sources of confusion traffic. Even in those configurations where confusion can theoretically be filtered out, eavesdropping tools may be susceptible to confusion. In fact, current eavesdropping tools appear to be especially vulnerable to even the simplest confusion techniques.

Acknowledgements

The authors would like to thank Harry Hoffman for his assistance configuring Bro and Snort for the experiments in Section 4. This work was partially supported by the NSF Cyber Trust Program under Grant CNS-0524047.

References

- [1] S. Bellovin, Wiretapping the net, *The Bridge*, vol. 20(2), pp. 21-26, 2000.
- [2] C. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin, Experimental quantum cryptography, *Advances in Cryptology – Proceedings of EUROCRYPT'90*, Springer-Verlag, Berlin-Heidelberg, pp. 253-265, 1990.
- [3] M. Blaze and S. Bellovin, Tapping on my network door, *Communications of the ACM*, vol. 43(10), p. 136, 2000.
- [4] E. Cronin, M. Sherr and M. Blaze, The Eavesdropper's Dilemma, Technical Report MS-CIS-05-24, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania, 2005.
- [5] E. Cronin, M. Sherr and M. Blaze, Listen too closely and you may be confused, *Proceedings of the Thirteenth International Security Protocols Workshop*, 2005.
- [6] C. Dickens, *A Tale of Two Cities*, April 1859.
- [7] R. Dingleline, N. Mathewson and P. Syverson, Tor: The second-generation onion router, *Proceedings of the Thirteenth Usenix Security Symposium*, pp. 303-320, 2004.
- [8] IEEE, IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE 802.3, 1985.
- [9] IEEE, Information Processing Systems – Local Area Networks – Part 4: Token-Passing Bus Access Method and Physical Layer Specifications, IEEE 802.4, 1990.
- [10] IEEE, IEEE Standard 802.11-1997 Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE 802.11, 1997.
- [11] ITU, Multifrequency Push-Button Signal Reception, Recommendation Q.24, ITU Telecommunication Standardization Sector, 1988.

- [12] R. Pang and V. Paxson, A high-level programming environment for packet trace anonymization and transformation, *Proceedings of the ACM SIGCOMM Conference*, pp. 339-351, 2003.
- [13] V. Paxson, Bro: A system for detecting network intruders in real time, *Computer Networks*, vol. 31(23-24), pp. 2435-2463, 1999.
- [14] J. Postel (Ed.), Internet protocol, Internet Engineering Task Force RFP 791, September 1981.
- [15] T. Ptacek and T. Newsham, Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection, Technical Report, Secure Networks, Inc., Calgary, Alberta, Canada, 1998.
- [16] M. Reiter and A. Rubin, Crowds: Anonymity for web transactions, *ACM Transactions on Information and System Security*, vol. 1(1), pp. 66-92, 1998.
- [17] R. Rivest, Chaffing and winnowing: Confidentiality without encryption (theory.lcs.mit.edu/~rivest/chaffing.txt), 1998.
- [18] U. Shankar and V. Paxson, Active mapping: Resisting NIDS evasion without altering traffic, *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pp. 44-61, 2003.
- [19] M. Sherr, E. Cronin, S. Clark and M. Blaze, Signaling vulnerabilities in wiretapping systems, *IEEE Security and Privacy*, pp. 24-36, November/December 2005.
- [20] G. Simmons, The prisoners' problem and the subliminal channel, in *Advances in Cryptology - Proceedings of CRYPTO'83*, D. Chaum (Ed.), Plenum Press, New York, pp. 51-67, 1983.
- [21] D. Song, fragroute (monkey.org/~dugsong/fragroute), 1999.

Chapter 17

ACTIVE TRAFFIC CAPTURE FOR NETWORK FORENSICS

Marco Slaviero, Anna Granova and Martin Olivier

Abstract Network traffic capture is an integral part of network forensics, but current traffic capture techniques are typically passive in nature. Under heavy loads, it is possible for a sniffer to miss packets, which affects the quality of forensic evidence.

This paper explores means for active capture of network traffic. In particular, it examines how traffic capture can influence the stream under surveillance so that no data is lost. A tool that forces TCP retransmissions is presented. The paper also provides a legal analysis—based on United States and South African laws—which shows that few legal obstacles are faced by traffic capture techniques that force attackers to retransmit data.

Keywords: Network forensics, active traffic capture, TCP retransmission

1. Introduction

Traffic capture has long been the mainstay of network forensics, providing the raw data which is analysed and dissected to reveal important information. This capturing or “sniffing” is an inherently passive process. Typically, the capturer places a network interface in “promiscuous” mode, and records each frame as it arrives. Passive capture can, in some circumstances, be a hindrance. Consider a situation where the capturer has missed a frame due to some error in the network or at the capturer. This gap in the record weakens the confidence in the information gleaned.

This paper proposes a new paradigm termed active capture, in which the capturer influences the communication stream under examination (within the bounds of the applicable protocols) to provide as clear a picture as possible of the event at hand. The influence could occur

Please use the following format when citing this chapter:

Slaviero, M., Granova, A., Olivier, M., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 215–228.

at almost any position in a network stack, and careful consideration is required to determine the optimal point for manipulating the traffic stream.

Active traffic capture is a controversial issue. Our purpose is to show that active capture is possible in a technical as well as legal sense. The technical and legal positions are discussed with regard to a hypothetical scenario. Alice, the IT security officer of Super Software Closed Corporation (SSCC), discovers that Eve, an aspiring cracker, is trying to penetrate the network entrusted to Alice. The picture becomes clearer and the evidence mounts as Alice passively monitors the network. At some point, the understanding of Eve's actions blurs because one of the packets involved in Eve's attack on the victim machine is lost. All Alice has to do to rectify the situation is to send a request for the lost packet to be resent. Since it is not unusual for such requests to come through, Eve would not be aware that she is being monitored.

Based on the hypothetical scenario, two questions arise: Is forcing Eve to resend lost packets technically feasible? Is it legal for Alice to engage in such behaviour and if so under what conditions?

This paper does not describe an active capture tool for network forensics. Rather, it discusses avenues for future research with the knowledge that active capture techniques may have legal standing. To this end, the technical discussion focuses on the placement of a capturer with respect to the traffic stream, a TCP implementation that forces data retransmission, and how traffic might be influenced during its capture.

The legal component discusses relevant United States (US) and South African (SA) legislation, including the US Electronic Communications Privacy (ECP) Act, the SA Constitution and related case law; the SA Electronic Communications and Transactions (ECT) Act; and the SA Interception Act.

The paper is structured as follows. The next section, Section 2, discusses technical issues related to active traffic capture using TCP retransmission. Section 3 examines the notion of "influence" and how it affects communications. Section 4 presents relevant legislation and case law. The final section, Section 5, presents the conclusions and avenues for future work.

2. Retransmission Technicalities

Packet-switched networks are relatively unreliable. Stone and Partridge [18] report Internet packet error rates as high as 1 in 1,100. Most data network implementations have many disparate modules; a small error or malfunction in just one module often creates error states in other

modules. Corruption is generally introduced at a physical level, in which signals on a medium are disrupted by faulty hardware or interference. Software errors are not unknown; the most common destination for erroneous packets is the bit-bucket, where the packets are simply discarded. In severely congested networks, data segments are lost when receivers (mostly routers) run out of buffer space to store incoming packets. Packets may also be lost due to unreachable routes or looping routes.

In the face of these obstacles, a retransmission strategy is used to ensure that data eventually gets to the correct receiver. Simply put, data is sent and resent until it is received.

Returning to the notion of disparate modules constituting a network, it is helpful to view them in a vertical fashion, with one module stacked on top of another. ISO's Open Systems Interconnect (OSI) model [8] is the most commonly used reference stack. Error states can occur at any of the seven layers in the OSI model.

Each layer in the OSI model can retransmit data if it detects errors; the function is not limited to certain modules. In the lower layers of the stack, collisions are common, and they are even expected in Ethernet networks. Collisions are quite normal because Ethernet is a multiple access medium: whenever a node wishes to send it does so. The OSI standard allows nodes to detect when multiple senders transmit at the same time. If collisions are detected, the senders halt and wait before retransmitting [1].

The Transmission Control Protocol (TCP) [11], which is higher in the stack, introduces reliability in IP [10] networks. (IP is the most prevalent packet-switched network protocol.) A receiving TCP tracks the bytes sent to it. If loss is detected, it informs the sending TCP, which then retransmits the lost sequence.

Finally, several higher-level protocols include facilities for retransmitting data. For example, the common email standard SMTP instructs mail servers to continually attempt to deliver mail until the destination acknowledges receipt. Thus, if a connection is reset while mail is being transferred, the originator will attempt to reconnect and resend the undelivered message [12].

All the retransmissions described so far are performed automatically. However, certain errors are not easily solvable by protocol specifications and these errors are usually reported to humans to investigate and solve, perhaps by manual retransmission. An example is the 504 error reported by web servers when a user attempts to visit a web page while the server is overloaded [4]. In this case, the retransmission occurs manually upon user input.

Returning to the attack scenario, assume that Alice wishes to track what Eve is doing for investigative or evidentiary purposes. Note that Alice is not a party to the communication, but she can record the communication. If a particular sequence of bytes in Eve's attack on a victim machine is lost to Alice, it might be possible to force the network stack on Eve's machine to resend the lost data. Retransmission can occur at various network layers, but the choice of layer at which to force retransmission is dependent on the type of attack.

If one of the lower stack layers is chosen, say the data-link layer, then a number of restrictions are placed on Alice. Foremost is overcoming the wide variety of link layer protocols. Under Ethernet, the goal of willingly forcing retransmission is extremely difficult, if not almost impossible. It would require the monitor to know that it has not read the packet while it is busy receiving the packet. This is because Ethernet interfaces only retransmit a frame if a collision occurred; there is no way to explicitly request retransmission and the monitor can only generate a collision. Also, this assumes the decision is made at the hardware level, while sniffers generally operate in software. In any event, such retransmission would require special hardware and is not explored any further in this paper. Consequently, higher layers of the stack must be examined.

Retransmission is very application specific at the highest layers of the network stack. If a monitor hopes to force retransmission, its software would have to process each higher layer protocol. This is a massive task given the vast array of protocols.

However, it is not unreasonable to conceive a situation where a monitor is interested in a specific protocol. A good example is a case where a blackmailer sends an email from an Internet cafe to a victim. Investigators suspect an individual, but may wish to gather more evidence, specifically time correlation data between when the email was sent and when the suspect was at the Internet cafe. Investigators could use a tool that fakes a bounced email to the suspect. The suspect, thinking that the original email was not received by the victim, then sends a second email. In such a situation, writing specialised software for a protocol might be worth the effort, but for general use, a compromise between the higher and lower stack layers would be necessary.

The techniques presented in the paper are applicable to reliable protocols, such as TCP, which guarantee that data arrives in the correct sequence. (UDP is a close cousin of TCP, but it does not provide reliability assurances.) However, TCP is agnostic towards the data it carries; the data is simply passed to the receiving application. TCP has a number of built-in facilities for retransmitting packets when losses are detected. Because TCP processing occurs in software on virtually all

TCP-capable nodes, it is possible to examine packets and determine if packets have been lost or read incorrectly *post factum*, and then request retransmission of the packets.

2.1 TCP Retransmission

This section discusses TCP retransmission. Background information about TCP is provided; readers are referred to [11] for additional details.

2.1.1 Sequence Numbers and Retransmission. Under TCP, the data being carried is viewed as a byte stream [11] by the sender $\mathcal{S}$, where each byte is numbered sequentially modulo 2^{32} . When a packet is sent, the number of the first data byte of the packet is carried along with the data. This is called the sequence number of the packet, which is denoted by *seq*.

A receiving TCP $\mathcal{R}$ sends periodic acknowledgements to $\mathcal{S}$ indicating θ , the point in the stream up to which all bytes have been received. Thus, if a packet is lost or corrupted, $\mathcal{R}$ simply keeps sending acknowledgements holding θ .

Vanilla TCP uses timeout queues to implement retransmission. When a packet is first sent, a copy is placed in a timeout queue. If an acknowledgement is received that covers the data in the packet on the timeout queue, it is deleted from the queue. Whenever a timeout occurs, the packet is resent, and it remains in the queue.

Jacobson [6] noted that a lost packet can be inferred by assuming that when three acknowledgements with the same sequence number arrive, the packet starting with that sequence number has not arrived. The sender can then retransmit the supposed missing packet. This improvement is documented in [17], and is incorporated in virtually all modern TCP implementations. Other acknowledgement schemes exist for TCP [5, 7], but support for these schemes is not as widespread.

2.1.2 Forcing Retransmission. At this point the gist of how TCP provides a high degree of reliability to upper layers should be clear. The problem for a third party who is attempting to capture a particular stream is that if, due to local conditions or network congestion, a particular packet does not arrive at the capturing station, a portion of the evidence might be lost.

TCP uses two methods for resending packets: queue timeouts and duplicate acknowledgements. The latter can be forced on the sender, while the former cannot be directly effected by a remote host. Therefore, we can choose one of two positions for the capturing party: (i) on the path between the two end points, which we term an “interceptor,” or

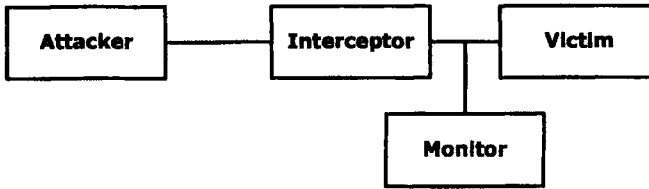


Figure 1. Placement of traffic capturing parties.

(ii) listening in on the communication at some point in the network, which we call a “monitor.” Note that the monikers, interceptor and monitor, respectively, correspond to the concept of being a party to the communication, or not being party to the communication, in terms of the South African Interception Act [16]. Figure 1 shows the placement of capturing parties in an attack scenario. Note that only one position needs to be assumed by the capturer.

An interceptor has more influence on a communication stream than a monitor. This is because the stream passes through the interceptor, but only past the monitor. The interceptor can perform transformations on the stream, while the monitor can only react to packets as they go by. Also, an interceptor can force a retransmission by blocking the acknowledgement from the receiver or altering the sequence number. When the associated timeout on the sender expires, the packet is resent. A drawback of interception is the overhead and possible delays that are created because every packet must pass through the interceptor and be examined by it.

The situation for the monitor is more tenuous. Direct action must be taken to force a retransmission and such an action must be expeditious. The rationale is simple: the monitor has to send three duplicate acknowledgements to the sender before the receiver can send one legitimate acknowledgement. It may seem unreasonable to presume this to be possible; however, such a monitor is generally placed at the ingress points to a network. Note that equipping a monitor with more network resources than the receiver reduces the likelihood of the monitor having to force retransmissions since, if a packet is lost, it will probably be lost at the receiver, which will send legitimate duplicate acknowledgements.

The retransmission strategies framed thus far are based on the fact that the capturing party knows when a packet has gone missing. It is easy for the capturer to determine if packets have been reordered.

Specifically, the capturing party $\mathcal{C}$ stores the sequence number seq for each $(\mathcal{S}, \mathcal{R})$ pair. When a packet P arrives from $\mathcal{S}_i$ destined for $\mathcal{R}_k$, $\mathcal{C}$ checks if $(\mathcal{S}_i, \mathcal{R}_k).seq = P.seq$. If it fails, then the most $\mathcal{C}$ can deduce

1. 23:15:36.920548 IP eliot.1092 > fornax.ssh: P 1368050088:1368050089(1)
ack 3457346665 win 64218
2. 23:15:36.920698 IP fornax.ssh > eliot.1092: . ack 0 win 64218
3. 23:15:36.920708 IP fornax.ssh > eliot.1092: . ack 0 win 64218
4. 23:15:36.920724 IP fornax.ssh > eliot.1092: . ack 0 win 64218
5. 23:15:36.921028 IP eliot.1092 > fornax.ssh: P 0:1(1) ack 1 win 64218
6. 23:15:37.052875 IP fornax.ssh > eliot.1092: . ack 1 win 5840

Figure 2. Packet trace.

is that a packet has arrived out of order. Checking for packet loss, however, requires an increase in storage space as the last few sequence numbers seen must be stored along with the times of arrival to determine if they have exceeded the Maximum Segment Lifetime. We ignore these complications and treat reordering as packet loss. This simplification is not unreasonable because it is not common to have many reordered packets in a single connection [3].

2.2 Proof-of-Concept Tool

A proof-of-concept tool that forces TCP retransmissions was developed. The tool uses the `libpcap` library [19] for packet capture and raw sockets to transmit forged packets. It can force retransmissions of packets that match certain criteria (using `tcpdump` expressions), enabling specific connections can be targeted.

The test network comprised three machines organised into two networks. The machines `eliot` and `narthex` constituted one network, and `fornax` the other. The machine `eliot` had retransmissions forced on it, `narthex` was the monitor, and `fornax` was the client. The two networks were linked by a slow line to exaggerate the time intervals between forged packets and legitimate packets.

Figure 2 displays a packet trace recorded on `narthex`. The machine `eliot` sends one byte of data to `fornax` in the first packet. The monitor, `narthex`, forges three packets from `fornax` to `eliot` (Steps 2–4). The acknowledgement number in the forged packets is unchanged (shown as 0), so `eliot` retransmits the packet in Step 5. Finally, the legitimate packet arrives from `fornax`, acknowledging the first packet.

Of particular interest in the packet trace are the time intervals between events. The creation and transmission of forged packets occurred in $176\mu\text{s}$, and the forced response arrived $304\mu\text{s}$ later. As noted previously, a slow link was purposely used between `fornax` and `eliot`; thus, the legitimate packet arrived 132ms after the initial data was sent. This simple test shows that a relatively unsophisticated tool needs a few hundred microseconds to enact the retransmissions.

3. Influence on Communications

A key question regarding traffic capture is whether the communication is unduly compromised by the capturing process. If it turns out that, when monitoring or intercepting a stream, data instead of only control packets were injected into the traffic, then the evidentiary value would be significantly reduced.

It can be argued that by merely plugging into the spanning port of a switch, additional load is created, which could affect the timing of a monitored conversation. The methods proposed in the previous section assume a much more proactive approach to capturing traffic, which certainly is a form of influence. The burden on the monitor is then to show how the influence, whatever it was, did not create, alter or delete data. Since the TCP retransmission technique uses control packets, which are part of the protocol, and data being transferred is not altered in any way, such influence is legitimate.

To verify that no undue influence has taken place, packet logs of the entire communication should be recorded. The logs would show where packets were inserted into the stream and what the effects were.

Of course, the capturer should follow standard investigative policies and practices that are reasonable and demonstrably reliable. The discussion of this issue is outside the scope of this paper.

Although our forced retransmission technique is limited to specific protocols, it is a first step towards active monitoring, which creates numerous opportunities for exploration. Such exploration must be carried out very carefully, otherwise the evidence collected would be inadmissible in court. The next section, which examines two legal systems, shows that merely using active techniques is not grounds for discounting evidence.

4. Legal Framework for Computer Interception

The legality of forcing network retransmissions depends on the laws of the country where retransmissions are attempted. This section focuses on the United States and South African legal regimes.

4.1 United States Framework

Two main acts constitute the U.S. legal framework that governs the interception of and access to information: (i) the Electronic Communications Privacy (ECP) Act (18 U.S.C. § 2510-2521), and (ii) the Stored Communications Act (18 U.S.C. § 2701-2707). For the purposes of this paper, however, only the ECP Act is relevant.

The ECP Act applies to all electronic communications, including the use of e-mail, cell phones, satellite communications and computer-to-computer communications [20]. As a general rule, any interception or disclosure of electronic communication as well as the use or procurement of equipment to intercept such electronic communication is prohibited (18 U.S.C. § 2511(1)) and any violation is levied a fine of \$500 or higher. The few exceptions to the rule are: consent to interception, interception in the course of performing duties at the workplace, and court orders authorising such conduct (18 U.S.C. § 2511(2)).

Returning to the Super Software Closed Corporation (SSCC) scenario, it is certain that Eve would not consent to Alice intercepting the attack on SSCC's network. Neither would she consent to a retransmission request. Furthermore, it may be difficult for Alice to obtain a court order that authorises these activities.

There should, however, be no doubt that if Alice resorts to forcing network retransmissions to compile a complete forensic report on Eve's unlawful network penetration, she would definitely be acting within the normal course of her duties as SSCC's IT security officer. As such, Alice's conduct would be legal and Eve would have no recourse against either Alice or SSCC (18 U.S.C. § 2511(2)(i); also see *Quigley v Rosenthal* 327 F.3d 1044 (10th Cir. 2003)).

4.2 South African Constitution

The 1996 Constitution of the Republic of South Africa [14] radically changed the notion of privacy that existed in non-democratic South Africa. Privacy, as a right, was entrenched by Section 14 of the Constitution, and it includes the right to privacy in communications [2].

But this right—like others in the Bill of Rights [14]—is not absolute. There are always competing rights that need to be weighed against each other. In our scenario, it would be, firstly, Eve's right to privacy as opposed to the right to privacy of Super Software Closed Corporation (SSCC). Secondly, SSCC's right to privacy would have to be balanced with Eve's right to a fair trial as envisaged in Section 35(5) of the Constitution. Section 35(5) provides that: "Evidence obtained in a manner that violates any right in the Bill of Rights must be excluded if the admission of that evidence would render the trial unfair or otherwise be detrimental to the administration of justice." The right to fair trial is important here because Alice would in almost all instances request that packets be retransmitted to obtain more evidence for possible legal action against Eve.

Finally, the right to privacy is, in any event, subject to the general limitation clause contained in Section 36 of the Constitution. To this end, the ECT Act and the Interception Act (discussed below) would have to be interpreted to comply with the said requirements. The two acts are considered after the discussion on case law related to the right to privacy, which follows.

4.3 Case Law

During the pre-1996 Constitutional era, two important cases were decided about the right to privacy. First, in *S v I and Another* (1976 (1) SA 781 (R, A.D.)), the court held that the invasion of privacy, which was reasonably necessary, solely with a *bona fide* motive to obtain evidence of adultery was legal. In our scenario, the limitation of Eve's right to privacy would definitely tip the scale in favour of the entitlement to protect the privacy of SSCC. This would make evidence obtained through forced retransmissions legal and admissible in criminal proceedings against Eve.

A leading civil case, decided by the Appellate Division of the Supreme Court, the highest court in South Africa at the time, was *Financial Mail (Pty) Ltd. and Others v Sage Holdings Ltd. and Another* (1993 (2) SA 451 (A)). In this case, the court held that the right to privacy applied to natural persons as well as to organisations. The court decided that the facts justified a conclusion that, in light of contemporary *boni mores* (good morals) and the general sense of justice of the community, the company's right to trade and carry on its business without wrongful interference from others deserved proper protection under the law within the ambit of the right to privacy.

In 2004, the Cape High Court in *Huey Extreme Club v McDonald t/a Sport Helicopters* (2005 (1) SA 485 (C)) held, with reference to Section 14 of the Constitution, that an invasion of privacy would be justified only where the *boni mores* of the community dictated that the limitation is reasonable. This is in light of the interests of an organisation that were sought to be protected. Clearly, IT security for a contemporary and successful business is as important as the interest of public safety for Mr. McDonald in the Cape High Court case. By analogy, this case serves as authority as to why Eve's privacy should be limited in favour of SSCC's interests.

In the light of the case law, it is evident that constitutional protection as enforced by the court today is present if and when required. More clarity, however, may be obtained by considering legislation that applies specifically to the hypothetical scenario.

4.4 ECT Act

The Electronic Communications and Transactions Act 25 of 2002 [15] covers the interception of communications, including computer-related communications, in a very general way. Section 86 of the ECT Act prohibits any interception of data, whether passive or active.

Before we discuss the Interception Act itself, one more reference needs to be made. Sections 86(3) and (4) of the ECT Act make it an offence, *inter alia*, to create and/or buy any software programs that help overcome security measures that are in place for protecting data.

4.5 Interception Act

The Regulation of Interception of Communications and Provision of Communication-Related Information Act 70 of 2002 [16] addresses direct and indirect communication. Transfer of data is classified as “indirect communication” in the Preamble of the Interception Act, since it takes place over a “telecommunication system” [13]. Interception, pursuant to this act, encompasses any acquisition and including the redirection of the flow of any communication through any means. Such wide scope of application is bound to cover the actions of Alice when she manually requests a packet to be retransmitted by Eve’s computer.

Two situations are covered by the Interception Act: (i) where Alice is a party to the communication in question, and (ii) where she is not. With respect to Figure 1, Alice is either an interceptor or monitor.

In the first situation, the law is clear: monitoring by Alice is allowed according to Section 4(1) [16]. This section provides that: “Any person, other than a law enforcement officer, may intercept any communication if he or she is a party to the communication, unless such communication is intercepted by such person for purposes of committing an offence.” As for the second set of facts, in the case of a monitor, the burden of proof is on Super Software Closed Corporation (SSCC) to demonstrate that the interception of indirect communication did, in fact, comply with the requirements of Section 6 of the Interception Act [16].

The first requirement is that interception must take place in the course of conducting normal business activities. Considering the facts here, it would entail providing proof that the interception falls within the powers of Alice who is charged with protecting SSCC’s network from intrusions. Secondly, SSCC would have to show that there was consent of the system controller, whether express or implied, to the interception. Thirdly, the interception would have to be done for a legitimate purpose as defined in Section 6 of the act. In particular, the active gathering of information would have been done to: (i) establish the existing facts (e.g., of the very

intrusion), or (ii) investigate or detect unauthorised use of the network (i.e., to gather evidence for possible legal action against Eve).

5. Conclusions

This work should be of interest to the technical and legal communities. Forced retransmission is potentially a useful strategy for IT security officers and digital forensic investigators. The TCP retransmission technique based on duplicate acknowledgements enables data resending to be forced by a third party. Clearly, this technique can be very beneficial for investigative and evidentiary purposes.

It appears that manual requests for network retransmissions would be considered legal by United States and South African courts, and evidence retrieved by such means would be admissible in civil and criminal trials. IT security officers could, therefore, become more proactive in assisting law enforcement agencies and preventing illegal activities in United States and South African networks.

This approach to active traffic capture is limited to forcing the retransmission of individual TCP frames. Our current and future work is aimed at developing active capture techniques for other layers of the network stack.

Acknowledgement

The authors would like to thank the members of the ICSA Research Group for their comments.

References

- [1] ANSI, Information Processing Systems: Local Area Networks – Part 3, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, American National Standards Institute, 1992.
- [2] P. Bekker, T. Geldenhuys, J. Joubert, J. Swanepoel, S. Terblanche and S. van der Merwe, *Criminal Procedure Handbook (Sixth Edition)*, Juta and Company, Lansdowne, South Africa, 2003.
- [3] J. Bellardo and S. Savage, Measuring packet reordering, *Proceedings of the Second ACM SIGCOMM Workshop on Internet Measurement*, pp. 97-105, 2002.
- [4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, Hypertext transfer protocol – HTTP/1.1, *RFC 2616*, Internet Engineering Task Force, June 1999.

- [5] S. Floyd, J. Mahdavi, M. Mathis and M. Podolsky, An extension to the selective acknowledgement (SACK) option for TCP, *RFC 2883*, Internet Engineering Task Force, July 2000.
- [6] V. Jacobson, Congestion avoidance and control, *Proceedings of the ACM SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 314-329, 1988.
- [7] M. Mathis, J. Madhavi, S. Floyd and A. Romanow, TCP selective acknowledgement options, *RFC 2018*, Internet Engineering Task Force, October 1996.
- [8] ISO, Information Processing Systems – OSI Reference Model – The Basic Model (ISO 7498-1:1994), International Organization for Standardization, 1994.
- [9] V. Paxson, End-to-end Internet packet dynamics, *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication*, pp. 139-152, 1997.
- [10] J. Postel, Internet protocol, *RFC 791*, Internet Engineering Task Force, September 1981.
- [11] J. Postel, Transmission control protocol, *RFC 793*, Internet Engineering Task Force, September 1981.
- [12] J. Postel, Simple mail transfer protocol, *RFC 821*, Internet Engineering Task Force, August 1982.
- [13] Republic of South Africa, Telecommunications Act (Act 103), 1996.
- [14] Republic of South Africa, Constitution of South Africa (Act 108), 1996.
- [15] Republic of South Africa, Electronic Communications and Transactions Act (Act 25), 2002.
- [16] Republic of South Africa, Regulation of Interception of Communications and Provision of Communication-Related Information Act (Act 70), 2002.
- [17] W. Stevens, TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms, *RFC 2001*, Internet Engineering Task Force, January 1997.
- [18] J. Stone and C. Partridge, When the CRC and TCP checksum disagree, *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 309-319, 2000.
- [19] TCPDUMP (www.tcpdump.org).

- [20] J. Winn and B. Wright, *The Law of Electronic Commerce (Fourth Edition)*, Aspen Publishers, New York, 2005.

Chapter 18

LOGICAL TRAFFIC ISOLATION USING DIFFERENTIATED SERVICES

Tinus Strauss, Martin Olivier and Derrick Kourie

Abstract This paper proposes a scheme in which the differentiated services field of IP headers is used to logically isolate network traffic for forensic purposes. The scheme is described and two example scenarios are presented to illustrate its utility. The scheme, which is based on standard networking technology, helps achieve isolation without additional network infrastructure. Moreover, the scheme is relatively easy to implement in an existing differentiated services network. The paper also discusses key design and configuration challenges that must be addressed in a successful implementation.

Keywords: Network forensics, differentiated services, traffic isolation

1. Introduction

Genge [8] describes the dilemma that faces first responders in classical (probably non-digital) events:

“The first person on the scene is immediately confronted with a number of considerations: victims who may be in need of immediate attention, witnesses ready to melt away at the first opportunity, the possibility of further criminal activity, the responsibility of preserving whatever evidence might be remaining and securing a crime scene while maintaining safe corridors for emergency personnel. This person must weigh all these needs and make immediate decisions based on the situation. And every situation is, in some way, unique.”

Current network forensic procedures often do not have the sophistication to handle real-world incidents. Frequently, a simplistic approach is prescribed: unplug the compromised network host. While this may prevent further damage to the network, it does not necessarily preserve evidence that might remain. Also, it prevents the network from supporting the operations of the organisation.

Please use the following format when citing this chapter:

Strauss, T., Olivier, M., Kourie, D., 2006 in International Federation for Information Processing, Volume 222. Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 229–237.

Deciding whether or not to isolate a compromised host from a network [4] is a question of balancing the requirements of a forensic investigation and the necessity of maintaining the availability of resources. Disconnecting the host from the network isolates it completely, allowing the damage caused by the malicious party to be assessed, documented and possibly corrected. The state of the host at the point of its removal from the network embodies all the evidence available on the system. The evidence might not be enough to convict the perpetrator, or the criminal activity up to that point might not have been severe enough to warrant the cost and effort of prosecution.

On the other hand, if the host is left connected to the network, the perpetrator might be unaware of the fact that his activities have been discovered and continue to engage in them. These activities could then be recorded, resulting in a stronger case against the suspect. The activities may, however, cause further damage to network assets.

In a true networked environment this problem is exacerbated by the fact that an incident is likely to involve multiple hosts and the entire network may have to be unplugged. Casey [4] suggests following approach to address these concerns:

“However, when the system is a critical component of a network, it may be necessary to involve network administrators to reconfigure a router or firewall, partially isolating the system but permitting vital connections to enable an organisation to remain in operation.”

While it is possible to manually configure, or even build, custom solutions for networked systems that allow the type of forensic isolation alluded to above, a more general solution is required. The fact is that one does not necessarily know where and when an incident will happen, and a solution that can be deployed in a significant number of cases, once an attack is underway or has just occurred, is required. It is therefore necessary to achieve isolation based on technologies that are already deployed and that are well understood by network administrators.

This paper presents a scheme that provides a balance in that the host is not removed from the network, but a variable degree of isolation is achieved through the logical separation of relevant packets from the rest of the traffic. This is achieved by using the Differentiated Services scheme, a standards-based technology commonly supported in routers and other networking equipment.

The degree of isolation depends on the nature of malicious activity. If the compromised host is used as a platform for further crimes, this limited isolation with surveillance is ideal since the activities can be monitored. If the result of allowing the suspect to continue with his activities becomes too costly, the node and the suspect can be disconnected from

the network. The captured network traffic can then be analysed and presented as evidence [3, 6].

The next two sections discuss the concept of Differentiated Services and the logical isolation scheme based on Differential Services. The final section, Section 4, concludes the paper and identifies avenues for future work.

2. Differentiated Services

The Differentiated Services (Diffserv) scheme [1] was devised as a scalable approach to service differentiation in IP networks. The scheme examines the DS field [10] in the IP header and, based on the value in the field, a packet is treated in some predefined manner at each hop on the path to its destination. The value of the DS field is referred to as the differentiated services codepoint (DSCP).

Packets are marked, i.e., assigned a DSCP, according to certain rules and conditions. The marking decisions could be based on temporal properties of the arriving packet stream or on something as simple as the source of the arriving packet.

One reason for the scalability of the Diffserv scheme is that sophisticated functions such as classification and marking are only performed at the boundary of a Diffserv-capable network; the interior nodes simply use the (DSCP) marks to determine how to treat packets. Another reason is that packets are aggregated into groups or classes with the same DSCP. Traffic is considered in aggregates that require the same treatment: flows are not treated individually.

The IETF has specified two per-hop behaviour groups: the Expedited Forwarding (EF) per-hop behaviour group [5, 7] and the Assured Forwarding (AF) per-hop behaviour group [9]. EF per-hop behaviour provides a building block for low delay and low loss services; the intent is to ensure that suitably marked packets encounter short or empty queues in the forwarding path.

AF per-hop behaviour provides a means for a network operator to offer different levels of forwarding assurance to packets. Four AF classes are defined and each class is allocated resources, such as buffer space, in each of the nodes in the Diffserv network. The packets belonging to each of the four classes are thus, in a logical sense, isolated from each other. Within each class, packets are marked with one of three drop precedence values. The precedence value assigned to a packet depends on whether or not the traffic stream is within the agreed-upon profile. If the traffic stream exceeds the profile, packets are given a higher drop precedence. Packets with higher drop precedence are discarded with a higher probability than

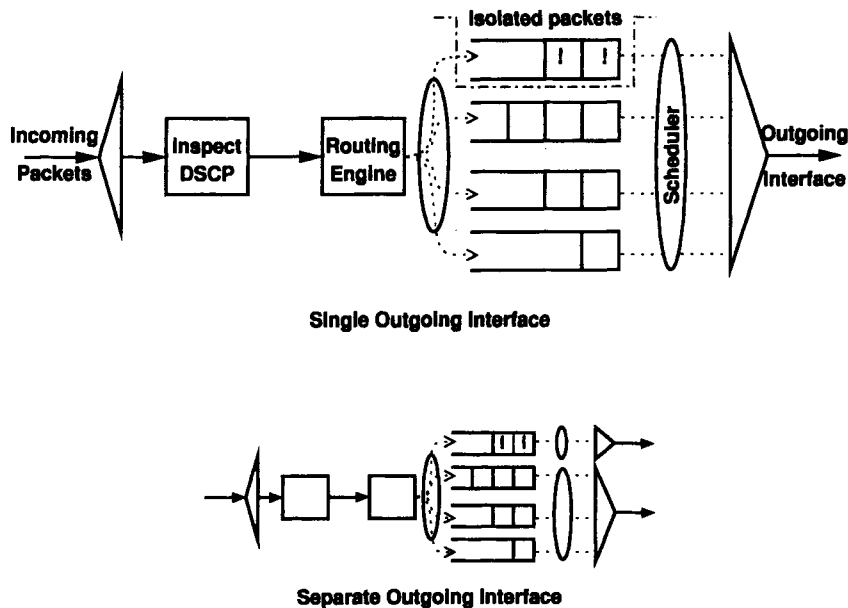


Figure 1. DSCP-based isolation inside a router.

packets with lower precedence. In this way, a network provider is able to offer different levels of service to its subscribers.

Since the provider of the Diffserv network, be it an organisation or an ISP, is free to implement its own marking policies, the organisation may reserve a set of DSCPs to assign to packets that are of forensic interest. Such a scheme is described in the next section.

3. Employing Diffserv for Logical Isolation

The idea behind the logical isolation scheme is simple: assign a DSCP to a packet that is deemed to be of forensic interest. The packet is thus identifiable throughout the Diffserv domain as associated with a specific criminal activity or surveillance effort. The marked packets are then placed into dedicated queues, which logically isolate them from regular packets.

Figure 1 provides a graphical representation of the scheme within a single Diffserv-capable router. When a marked packet arrives at an incoming interface of a router, its IP header (including the DS field) is inspected, and a decision is made about the relevant outgoing interface. The packet is then placed into the appropriate queue. If the packet is marked as forensically interesting (denoted by “!” in Figure 1), it is

placed in the dedicated queue; otherwise it is placed in one of the other queues. In this way, forensically interesting packets are logically isolated.

The scheduler can be configured to manipulate the rate at which forensic packets are serviced by the node. It is, therefore, possible to slow the traffic down.

Note that the routing decision may or may not take the DSCP into account. In Figure 1 (upper diagram), it is assumed that the routing is done independent of the DSCP. In the lower diagram of Figure 1, packets earmarked for forensic analysis are placed in a queue with a dedicated outgoing interface. This provides a greater level of physical isolation of packets.

If the DSCP is considered when making a routing decision, it is possible to route packets of forensic interest differently from regular packets. This enables the network operator to steer marked packets to certain points in the network where they may be captured for preservation and analysis. The router may be configured so that all packets matching the relevant DSCP—in addition to being forwarded to the outgoing interface—are copied and sent to a secondary interface where they are recorded.

3.1 Example Scenario

Figure 2 presents a network that implements the isolation scheme. The network is connected to the Internet, and two suspects (S1 and S2) are communicating with hosts H1 and H3, respectively. The network incorporates a marking station (MS), preservation station (PS) and management station (MGT). The marking station is responsible for marking packets; this function could be performed by an appropriately configured firewall or intrusion detection system (IDS). The preservation station collects and preserves network traffic for reconstructive traffic analysis. The management station configures and manages network elements.

Assume that host H1 is compromised and that the compromise is discovered. Instead of disconnecting H1 from the network, the investigators decide to keep it connected and monitor the situation. The Diffserv-enabled switch is configured (via the management station) to mark all packets entering the switch on the port connecting host H1. The packets are then isolated, and prevented from adversely affecting regular network traffic.

Figure 2 illustrates the case where the DSCP is also used to make a routing decision, since the packets are steered through the preservation station PS. The marking station MS at the boundary with the Internet marks all incoming packets destined to host H1 and these are steered to

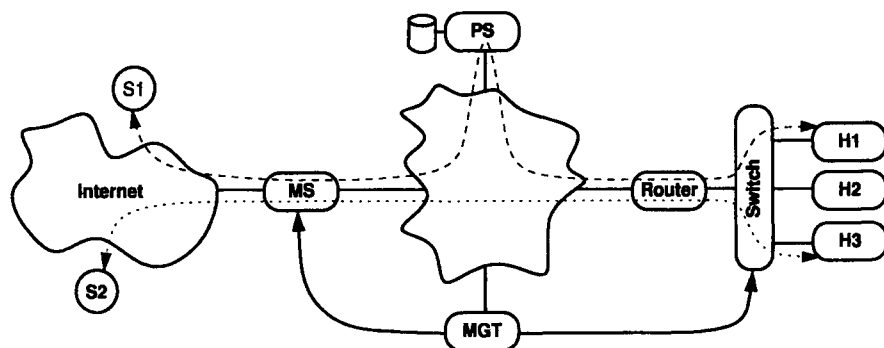


Figure 2. Example network.

the preservation station. All network traffic between S1 and H1 is now isolated and recorded for forensic analysis. Also, any traffic generated by H1 is marked and isolated, which protects the rest of the network infrastructure.

The scenario described is reactive in nature: a break-in was detected upon which a configuration change was made. It is also possible to have the IDS serve as a marking station. Traffic identified as being suspicious by the IDS is marked and automatically isolated. The traffic could then be collected or monitored in real time to determine if it is of forensic interest.

The communication between S2 and H3 in Figure 2 illustrates the case where the marked traffic is isolated but not steered to the preservation station. In this case, collection could occur through some other means.

3.2 Implementation Challenges

The isolation scheme is simple and elegant. There are, however, several challenges that must be addressed when implementing the scheme.

One challenge is deciding which packets to mark. The decision is simple if an investigation is already in progress: mark all packets headed to a certain destination or originating from some source. Alternatively, as in the example above, one might mark all the packets destined to the host as well as all the packets that enter the Diffserv-capable switch on the port that serves the compromised host. It is more difficult to mark traffic that is not yet associated with a crime, but which might be useful for forensic purposes.

The location of the marking station is also an issue. It would seem to be appropriate to place the station at a choke point in the network—at the Internet connection or at the server farm. An IDS or firewall

might be configured to mark packets that meet certain criteria instead of blocking them. Marking typically occurs at the boundary of the Diff-serv domain and not at the nodes inside the domain. Multiple marking stations could be implemented at the Internet gateway, server farm or elsewhere.

Since Diffserv provides different levels of service to different packets, it is necessary to decide which service levels should be granted to the marked traffic. Two of the levels at which to consider this issue are the engineering/provisioning level and the operational level. How much capacity should the operator dedicate to the isolated traffic? Should the operator be able to adjust the isolated traffic to slow it down to reduce damage or aid in real-time analysis? It seems appropriate to minimise the loss of isolated packets in order to preserve evidence.

The location of the preservation station should be considered carefully. Since all the relevant marked traffic passes through the preservation station, the location of the station can impact the load on the network. Resource bottlenecks could result if preservation stations are placed inappropriately. Note that multiple preservation stations could be positioned at suitable locations in the network.

The scheme described here is designed for a single Diffserv domain. It is possible to extend it to multiple domains if all the domains agree on the DSCPs to use for marking traffic.

Diffserv is based on traffic aggregates, not single flows. The scheme, therefore, isolates marked traffic from regular traffic, but it does not isolate individual traffic flows under investigation.

Note that the last marking station should remove the marks on packets when they leave the Diffserv domain. Otherwise the target of the investigation might be alerted about the surveillance effort.

3.3 Advantages

The principal advantage of the logical isolation scheme is that it does not require network operators to introduce additional technology. Of course, this assumes that the operators are already using Diffserv.

The scheme is extensible and easily modifiable. Since the scheme is based on Diffserv, which is defined by the IETF, it is a standards-based approach. This allows network operators to change equipment without performing major reconfigurations to achieve logical isolation. Moreover, the scheme is based on technology that is readily available in networking equipment. It is only a matter of configuring the Diffserv domain appropriately; this results in cost savings.

Diffserv is defined for both IPv4 and IPv6 packets, enabling the scheme to be used in networks running either IP version. Finally, since only marked traffic is captured, the scheme could aid in preserving privacy while enabling investigators to obtain evidence. This assumes, of course, that only appropriate packets are marked.

4. Conclusions

The Diffserv-based scheme provides for variable levels of logical isolation while balancing the requirements of forensic investigations and the necessity of maintaining the availability of resources. Since the scheme is built upon a flexible standards-based technology that is readily available, it is relatively easy and cost-effective to implement.

The scheme assumes that only relevant packets are marked for isolation. This is, however, not as simple as it appears because packets must be marked while a crime is being committed but before the crime has been detected. This is an important issue that needs further research.

Minimising the loss of isolated packets is also an issue. Resources must be reserved to accommodate these packets: reserving too few resources results in the loss of evidence; reserving too many leads to wasted resources. Guidelines must be developed for allocating resources.

The idea underlying the logical isolation scheme can be applied to other technologies, such as MPLS [11]. Labels could be used to isolate packets by routing along label-switched paths that are dedicated to packets of forensic interest. Another technology that could naturally allow for isolation is a provider-provisioned virtual private network (VPN) [2]. Since such a VPN isolates traffic and is under the network provider's control, it can function as a VPN for forensic purposes.

References

- [1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, An architecture for differentiated services, *RFC 2475*, December 1998.
- [2] R. Callon and M. Suzuki, A framework for layer 3 provider-provisioned virtual private networks, *RFC 4110*, July 2005.
- [3] E. Casey, Network traffic as a source of evidence: Tool strengths, weaknesses and future needs, *Digital Investigation*, vol. 1(1), pp. 28-43, 2004.
- [4] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*, Elsevier Academic Press, London, United Kingdom, 2004.

- [5] A. Charny, F. Baker, B. Davie, J. Bennett, K. Benson, J. Le Boudec, A. Chiu, W. Courtney, S. Davari, V. Firoiu, C. Klamaneck, K. Ramakrishnan and D. Stiliadis, Supplemental information for the new definition of the expedited forwarding per hop behavior, *RFC 3247*, March 2002.
- [6] V. Corey, C. Peterman, S. Shearin, M. Greenberg and J. van Bokkelen, Network forensic analysis, *IEEE Internet Computing*, vol. 6(6), pp. 60-66, 2002.
- [7] B. Davie, A. Charny, J. Bennett, K. Benson, J. Le Boudec, W. Courtney, S. Davari, V. Firoiu and D. Stiliadis, An expedited forwarding per hop behavior, *RFC 3246*, March 2002.
- [8] N. Genge, *The Forensic Casebook — The Science of Crime Scene Investigation*, Ebury, London, United Kingdom, 2004.
- [9] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski, Assured forwarding per hop behavior group, *RFC 2597*, June 1999.
- [10] K. Nichols, S. Blake, F. Baker and D. Black, Definition of the differentiated services field in the IPv4 and IPv6 headers, *RFC 2474*, December 1998.
- [11] E. Rosen, A. Viswanathan and R. Callon, Multi protocol label switching architecture, *RFC 3031*, January 2001.

Chapter 19

PASSIVE DETECTION OF NAT ROUTERS AND CLIENT COUNTING

Kenneth Straka and Gavin Manes

Abstract Network Address Translation (NAT) routers pose challenges to individuals and organizations attempting to keep untrusted hosts off their networks, especially with the proliferation of wireless NAT routers. Residential NAT routers also create problems for Internet Service Provider (ISP) taps by law enforcement by concealing network clients behind cable or DSL modems. This paper discusses the feasibility and limitations of methods for detecting NAT routers and counting the number of clients behind NAT routers.

Keywords: NAT routers, passive detection, client counting

1. Introduction

Network Address Translation (NAT) routers provide a convenient way to share a single IP address between several clients, but they can pose serious security risks [3, 8]. A NAT router connected to an Ethernet port in an office environment allows several—possibly untrusted—clients to access network resources. Some ISPs use MAC address filtering to keep multiple clients from using their services. However, commercial NAT routers can clone a client's MAC address. This MAC cloning allows a NAT router to masquerade as a client computer so that the connected network does not suspect other connected devices. In this sense, a NAT router can subvert many commonly used security protocols, e.g., MAC address authentication. Therefore, MAC address filtering is not an effective technique for detecting and mitigating NAT routers.

NAT router functionality is commonly combined with IEEE wireless fidelity standards 802.11b and 802.11g [2], allowing users with appropriately equipped computers or laptops to wirelessly access the Internet via a NAT router. Users without specialized training may not realize that

Please use the following format when citing this chapter:

Straka, K., Manes, G., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 239–246.

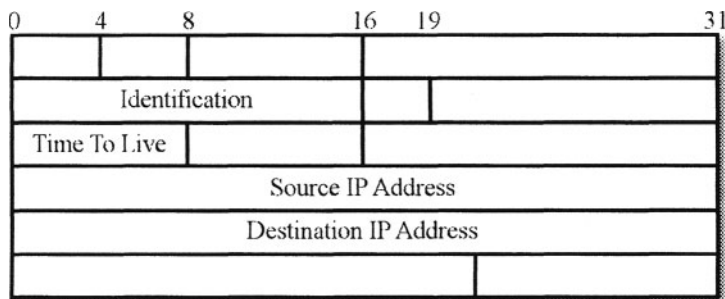


Figure 1. IP header fields relevant to detecting NAT routers and counting clients.

plugging a wireless NAT router into an Ethernet port with default settings in order to use a wireless laptop can create an insecure access point to the internal network [7]. A war-driver or other malicious entity outside the physical protection perimeter could access the internal network and launch an attack [4]. Finding and incapacitating rogue wireless NAT routers consume time and resources—detection usually involves “war-walking” with a wireless laptop to search for unauthorized wireless access points. Thus, it is often the case that rogue routers remain operational for extended periods of time.

In addition to leaving a network open to unauthorized clients, NAT routers effectively conceal the identity and number of connected clients. Therefore, ISP taps performed by law enforcement in computer crime investigations may not provide accurate information. Knowing the number and type of clients residing behind a router is useful for preparing a search warrant, as it may indicate the number of network clients to be seized and the level of technological sophistication of the suspect. Furthermore, detecting unauthorized NAT routers is essential to maintaining security in any enterprise network.

A NAT router shares a single Internet connection between several network clients by assigning a local, non-routable IP address to each network client [8]. The NAT router acts as a gateway when a client sends a packet beyond the NATed LAN (a remote server). When the NAT router receives the packet, it changes several fields (Figure 1). The most important is the packet’s source IP address, which is replaced with that of the router. This allows the response from the remote server to return to the correct world-routable IP address. In addition, the NAT router replaces the packet’s original source port with an arbitrary high-numbered port associated with the client’s source port. Upon receiving the response from the remote server, the NAT router checks the destina-

tion port against its table of client source ports. It then routes the packet to the appropriate client after recreating the original source port. This is necessary because a packet received by the router is addressed to the world-routable IP address, and without an additional demux key (the destination port on the inbound packet) it is unclear which client should receive the packet. When a new source port is used by any client for the next stream, it also translates to a new source port on the post-NAT router packet (usually the last translated source port plus one, regardless of which client created the new source port), showing a one-to-one relationship between original source ports and translated source ports. The source port progression apparent on the public side of the NAT router is almost indistinguishable from that of a directly-connected, solitary client.

The next section describes current methods for detecting NAT routers. The remaining sections discuss the feasibility and limitations of methods for detecting NAT routers and counting the number of clients.

2. Detection Methods

Although several methods exist to actively detect access points and NAT routers, they are less than ideal. The Network Mapper (nmap) can be used to perform TCP fingerprinting on network clients and determine which clients are running a router OS [5, 6]. This process, which is prone to false positives and false negatives, may take a long time for a large network and produces suspicious traffic that can trigger a local intrusion detection system (not a good attribute for a network defense tool). Another active method is similar in its approach and flaws. It uses Simple Network Management Protocol (SNMP) [10] scanning in combination with the `snmpwalk` tool to determine a client's OS [9].

Several methods exist to passively detect NAT routers. Examining the TTL (time to live) value of an IP packet can help determine if the packet has passed through a network device (NAT router) that decremented the TTL. In this case, the TTL value is one less than what is expected when the packet reaches the internal networking hardware.

Another method for detecting NAT routers and counting clients is to examine the id field in IP packet headers; this field is often used as an unofficial counter of outbound packets [1]. However, this method can result in false positives because some operating systems use the per-source-port counter as a global counter instead of the id field. Yet another method relies on the fact that input/output source ports assigned to a client computer by a NAT router are very similar to those the client computer would assign to its outgoing packets. For example, Mandrake 9.2 assigns

source port 32,885 to an outbound packet, which is changed to 32,774 by a NAT router. This progression is very similar to that of the client itself: it simply involves counting up from a relatively high port number as each new source port from a client is used. Examining the general numeric range of source ports may give some insight into the type of communicating device, and a sudden change in source port numbering may indicate a new device has been plugged into the port. Determining whether the device is a router would require a large sampling of NAT router behavior, which is beyond the scope of this paper. In any case, the method is unsuccessful unless a specific source port assignment scheme can be derived from the NAT router and the scheme is shown to be different from the scheme used by the client to assign source ports to outgoing packets.

The IP id and TTL counting techniques are somewhat unreliable when used individually. Our combined method uses simple TTL counting and IP id counting and grouping. In addition, it observes the treatment of outbound packets by the operating system, allowing for more accurate detection and client counting.

3. NAT Router Detection

Examining the TTL values of packets is one of the easiest and most reliable ways to detect NAT routers. Most systems have a default TTL value on outbound packets of 64 (Linux and Mac OS X) or 128 (Microsoft Windows). The TTL is decremented by one when packets pass through a NAT router before continuing to a piece of switching equipment. The presence of a TTL value other than 64 or 128 on a switch or router serving clients on an “edge” network segment would, therefore, indicate the presence of a NAT router—or at least some network device—between the client and the switch [1]. This method can be defeated by changing the default TTL of the client to 129 or 65 using `iptables` or a similar utility. However, most users are not aware of the specifics of IP packets, so the detection method should be quite effective.

IP id counting can also be used to detect NAT routers. If packets arriving at a network port exhibit multiple, distinct IP id sequences and each of these sequences continues across multiple source ports without interruption, a NAT router serving two or more clients is present at that network port. This point is discussed in more detail in the next section on counting clients.

4. Client Counting

Most operating systems use the id field in the IP header as a simple counter for outgoing packets. The number used to start the count is arbitrary: it can be below 100 or in excess of 50,000. Windows and Mac OS X use a global packet counting scheme, in which the counter is usually incremented by one (monotonically increasing) for each outgoing packet, regardless of the protocol, stream or source port. Some unexplained gaps in IP id counting have been observed on Windows and Macintosh machines, but the gaps are often exactly 55 (never more), which can be incorporated into a counting algorithm. The IP id field is typically left untouched by the NAT router when it translates the packet. As described in [1], assigning packets with sequential id fields to “groups” of similar id fields from past packets permits the estimation of the number of clients behind a NAT.

Linux systems count packets on a per-source-port basis, i.e., a new counter is started whenever a new source port is used for an outgoing communication. This presents a problem for the IP id counting method: a single Linux machine loading five simultaneous HTTP requests results in the use of five distinct source ports, and exhibits the same traffic pattern as five Macintosh machines, each loading one of the five HTTP requests. This results in the use of one source port per client, and there is no way to definitively determine which topology reflects reality.

Combining TTL analysis (128 for Windows, 64 for Mac/Linux, 127 for Windows behind a NAT and 63 for Mac/Linux behind a NAT) and IP id analysis provides a clearer picture of the number and types of clients served by a NAT router. If there is only one IP id sequence for all outgoing traffic, then there is only a single non-Linux computer at the network port. If there are multiple id sequences that cross source-port boundaries with minimal or no interruption, there are multiple non-Linux machines behind the NAT router.

Further information can be gathered based on the types of TTLs. A mix of TTL values of 63 and 127 indicates a mixed client pool that includes at least one Macintosh or Linux machine and at least one Windows machine. While Macintosh and Linux machines both have default TTLs of 64, they can be differentiated by their IP id behavior. Similarly, while Windows and Macintosh machines have similar IP id counting behavior, they can be differentiated by their default TTLs. The IP id counting method outlined in [1] works well in identifying the number of NAT clients only if there are no Linux machines behind the NAT router. In essence, the presence of a Linux machine reduces the reliability of passive client counting.

Other factors, besides the presence of Linux machines, can create problems for the IP id counting method. Some natively little-endian systems do not put the IP id field into network-byte order (e.g., in optional IP fields), but instead leave the id in little-endian: this causes the bytes to be read backwards by the sniffer, producing strange, non-sequential values. Fortunately, this can be easily managed by implementing a simple post-processing algorithm.

The IP id method may not work because not all operating systems use the IP id field. Even if one assumes that all the clients behind a NAT router are Macintosh or Windows machines and all are counting on a global packet basis, there are still several cases in which IP id method works poorly. For example, if there is a large amount of intra-LAN or subnet traffic, there are large gaps in the IP id sequences because the intranet traffic does not reach the post-NAT sniffer. This can result in an incorrect (higher) count of the number of clients. Furthermore, if IP id values from two or more hosts are similar and near in time, it is difficult to determine which packets belong to which client [1]. This problem may be mitigated by careful real-time tuning of time tolerance and sequence tolerance. Time tolerance is the maximum time allowed between two similarly numbered packets belonging to the same sequence. Sequence tolerance is the maximum distance between IP id numbers from the last packet received that is considered to be part of a sequence.

5. Application-Level Techniques

In addition to observing the relevant IP header fields of packets, examining application-level packet patterns can help detect NAT routers and count their clients. For example, a computer connected to an always-on Internet connection often has its email client automatically check for new mail every so often. If packet sniffing reveals bursts of Post Office Protocol (POP) traffic at 4, 6, 11, 13, 14 and 16 minutes, one could surmise that two email clients, most likely on two separate computers, are automatically checking for email at 5 minute intervals: one client at 4, 9 and 14 minutes and the other at 6, 11 and 16 minutes (see Figure 2). This hypothesis can be reinforced by checking the user names submitted to the POP server in each burst within POP "Request: USER (user-name)" packets. If there are two significantly different user names at the predicted times, there is a high probability that the traffic is caused by two separate computers behind a NAT router. However, if encryption is used (e.g., SSL or IPsec), such packet sniffing will probably not yield the desired information.

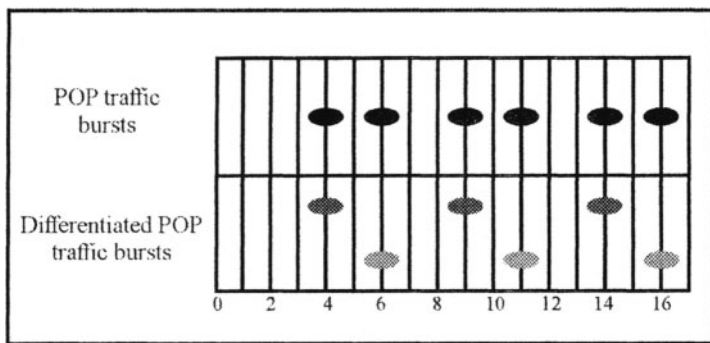


Figure 2. POP traffic bursts viewed as composed and differentiated by user name.

Note that this concept can be applied to many network activities that are likely to originate from a single computer on a periodic basis. This includes time server synchronization, operating system update checks (e.g., Windows Update) and application update checks (e.g., Norton Utilities' Live Update).

6. Conclusions

The best way to detect NAT routers is to look for numerically or logically odd TTLs on packets leaving the network port in question. The small number of packets and limited correlation required for this method to succeed makes it especially powerful. While this method is defeated by manipulating the default TTL values on client machines, it should still be able to detect the vast majority of rogue NAT routers.

IP id counting is effective at detecting NAT routers and counting the number of clients. However, research has shown that this technique occasionally works only because of the inconsistent manner in which different operating systems utilize the id field. When a per-source-port id counting machine is introduced behind a NAT router, the accuracy of the IP id method for counting hosts—or even detecting NAT routers—is greatly reduced.

Examining altered source ports is one of the least effective methods for detecting NAT routers because the altered ports are often similar to the source ports used initially by a client. Whenever a new source port is used by a client, the NAT router typically increments the most recent altered source port by one to use as the next altered source port. This monotonically increasing source port behavior is identical to that of most client operating systems. A large, sudden jump in source port

numbers not resulting from a numbering wrap-around indicates either a machine reset or a new network device connected to that port.

It is important to note that the detection and counting methods discussed in this paper assume that the NAT routers and operating systems tested are representative of the general population of NAT routers and operating systems. This assumption must be reviewed periodically to ensure its validity and the validity of methods that rely upon it. Furthermore, the methods themselves are not robust and should be audited regularly using supplementary techniques, such as physically checking all network ports.

References

- [1] S. Bellovin, A technique for counting NATed hosts, *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, pp. 112-208, 2002.
- [2] B. Crow, I. Widjaja, J. Kim and P. Sakai, IEEE 802.11 wireless local area networks, *IEEE Communications Magazine*, vol. 35(9), pp. 116-126, September 1997.
- [3] K. Egevang and P. Francis, The IP Network Address Translator (NAT), *RFC 1631*, May 1994.
- [4] A. Etter, A guide to war-driving and detecting war-drivers, SANS Infosec Reading Room, SANS Institute, Bethesda, Maryland (www.sans.org/rr/whitepapers/wireless/174.php), 2002.
- [5] R. Farrow, System fingerprinting with nmap, *Network Magazine* (www.itarchitect.com/article/NMG20001102S0005), November 2000.
- [6] Fyodor, Remote OS detection via TCP/IP stack fingerprinting, *Phrack Magazine*, vol. 8(54), December 1998.
- [7] J. Park and D. Dicoi, WLAN security: Current and future, *IEEE Internet Computing*, vol. 7(5), pp. 60-65, 2003.
- [8] L. Phifer, The trouble with NAT, *Internet Protocol Journal*, vol. 3(4), pp. 2-13, 2000.
- [9] S. Russell, Detecting and locating rogue access points (www.ee.iastate.edu/~russell/cpre537.s06/Report-Example.pdf), 2003.
- [10] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley, Reading, Massachusetts, 1998.

Chapter 20

ANALYSIS OF WEB PROXY LOGS

B. Fei, J. Eloff, M. Olivier and H. Venter

Abstract Network forensics involves capturing, recording and analysing network audit trails. A crucial part of network forensics is to gather evidence at the server level, proxy level and from other sources. A web proxy relays URL requests from clients to a server. Analysing web proxy logs can give unobtrusive insights to the browsing behavior of computer users and provide an overview of the Internet usage in an organization. More importantly, in terms of network forensics, it can aid in detecting anomalous browsing behavior. This paper demonstrates the use of a self-organising map (SOM), a powerful data mining technique, in network forensics. In particular, it focuses on how a SOM can be used to analyse data gathered at the web proxy level.

Keywords: Network forensics, web proxy logs, self-organising map, data analysis, anomalous behavior

1. Introduction

The Internet provides offenders with communication capabilities that did not exist previously. Internet-related crimes are on the rise and Internet abuse by employees is becoming routine.

Network forensics involves capturing, recording and analysing network audit trails to discover the source of security breaches and possible malicious activity [19]. It is becoming increasingly practical to archive network traffic and analyse the data as necessary [6].

A fundamental goal in network forensics is to gather evidence. Evidence can be gathered from various sources depending on the unique nature of the investigation. It can be collected at the server level, proxy level or from several other sources. For example, at the server level, evidence can be obtained from web server logs that record the browsing behavior of site visitors. Furthermore, evidence can be also gathered

Please use the following format when citing this chapter:

Fei, B., Eloff, J., Olivier, M., Venter, H., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S. (Boston: Springer), pp. 247–258.

from usage data provided by packet sniffers that monitor network traffic coming to a web server.

This paper deals with network forensics—more specifically, the analysis of web proxy data. Analysing data on a web proxy—as opposed to data on a single computer or on multiple computer systems—is significant. Users can delete traces of their Internet behavior from their computer systems. On the other hand, web proxy data pertaining to URL requests made by users is generally accessible only to network administrators and forensic investigators.

Another benefit is that investigators can focus on a single point in the network topology, which saves time that might be crucial to the investigation. For example, they can focus on employees in an organisation who access web sites that promote illegal activities. Since it is not necessary to seize employees' computer systems for evidence recovery, the investigation can be performed without the employees knowing that they are being investigated.

This paper demonstrates how a self-organising map (SOM) [11, 12], a powerful data mining technique, can be used in network forensics to analyse web proxy data. A SOM can reveal interesting patterns, and also serve as a basis for further analysis of the data gathered at the web proxy. More importantly, it can aid in detecting anomalous browsing behavior and in recovering digital evidence.

Sections 2 and 3 provide background information and an overview of SOMs. Section 4 demonstrates the use of a SOM to analyse web proxy data. The final section, Section 5, summarises the conclusions.

2. Background

While computer forensics goes as far back as 1984 [20], network forensics was introduced only in the early 1990s [19]. In general, computer forensics deals with data in a single computer system [22, 26]. Network forensics deals with data that may reside on computers, routers, firewalls, web servers, web proxies and other devices in one or more networks [4].

A crucial part of network forensics is to gather evidence. For example, when an attacker attacks a network, the attack traffic usually goes through a router. As a result, important evidence may be found by examining network logs.

Web proxy logs provide valuable evidence. The purpose of a web proxy is to relay URL requests from clients to a server, receive the responses from the server and send them back to the appropriate clients [17]. The web proxy acts as a gateway between the Internet and browsers on a local network.

Web mining is the extraction of interesting and useful knowledge, as well as implicit information from activities related to the World Wide Web [1]. It is categorised into three main areas: web content mining, web structure mining and web usage mining [16].

Similar to web mining [8, 14–16, 18], analysing web proxy logs can assist forensic investigators (or network administrators) in understanding the browsing behavior of computer users and in providing them with an overview of Internet usage in an organisation. Furthermore, it can assist them in gathering evidence left behind by suspects. Such evidence may involve excessive Internet usage for non-work purposes, or access to web sites promoting pornography and other illegal activities.

Considerable work has been done in the area of web usage mining [1, 2, 5, 10, 24]. In general, web usage mining involves three phases: data pre-processing, pattern discovery and pattern analysis. Web usage mining seeks to reveal knowledge hidden in web server log files, including statistical information about site visitors, and the preferences, characteristics and navigational behavior of computer users.

The self-organising map (SOM) approach used in this paper is fairly similar to web usage mining. SOMs have been used by researchers in a wide variety of fields [3, 7, 21, 25], but they have rarely been used in digital forensics.

SOMs have been used to cluster web pages according to user navigation behavior [23] and to organise web documents based on their content [13]. Our work uses a SOM to cluster, visualise and analyse web usage data gathered at a web proxy. Multi-dimensional data in web proxy logs is transformed by a SOM to two-dimensional data, which can be visualised and analysed more efficiently by forensic investigators.

3. Self-Organising Maps

The self-organising map (SOM) [11, 12] is a neural network model that has been widely used to cluster and visualise high-dimensional data. Clustering attempts to group data with similar characteristics [27]. Visualisation is the process of mapping complex data to a graphical representation to provide qualitative notions of its properties.

A SOM is used to map high-dimensional data onto a low-dimensional (typically two-dimensional) space. It consists of two layers of units (neurons), the input layer and the output layer. Each unit in the input layer, which represents an input signal, is fully connected with units in the output layer. The output layer forms a two-dimensional grid of units, where each unit represents a unit of the final structure.

A SOM employs unsupervised competitive learning, in other words, the learning process is entirely data driven and units in the output layer compete with one another. The learning process involves two steps: identifying the winning unit and updating unit weights. When an input pattern is presented to the input layer, the winning unit in the output layer is the one whose weights are closest to the input pattern in terms of the Euclidian distance [9]. After the winning unit is determined, the weights of that unit and its neighboring units are adjusted. The learning process continues until the SOM produces acceptable results or a pre-set limit is reached on the number of iterations.

The effect of the learning process is to cluster similar patterns. An additional step (determination of cluster boundaries) is required to visualise clusters. This is done by calculating the unified distance matrix [9]. The size of a cluster is the number of units allocated to the cluster.

One of the advantages of a SOM is its ability to manifest possible correlations between different dimensions of input data in component maps [28]. Each component map displays the spread of values in a particular dimension. Correlations are revealed by comparing component maps.

A SOM is also ideal for association and classification. Association seeks to identify correlations in data. Classification maps a data item into one of several predetermined classes. Network forensic investigations typically involve the analysis of enormous amounts of data. The ability of a SOM to support the clustering, correlation, association, classification and visualisation of multi-dimensional data make it an attractive tool for network forensics.

4. Analysing Web Proxy Data

We used a SOM to analyse web proxy logs for twenty computer users in an organisation. The proxy logs, which were generated by a Squid Proxy [29] over a period of one month, contained data pertaining to 374,620 HTTP requests. A Squid Proxy log has the following format:

```
time:etime:client:log-tag:size:request:url:user:userid:hierarchy
```

Our approach to analysing web proxy data in support of network forensics is fairly similar to web usage mining [1, 2, 5, 10, 24]. The following subsections discuss the three main phases: data pre-processing, pattern discovery and pattern analysis.

4.1 Data Pre-Processing

Data pre-processing is concerned with data cleaning and data transformation. The goal of data cleaning is to remove irrelevant information.

Data transformation, on the other hand, converts raw data items into structured information.

Eliminating irrelevant data simplifies SOM learning and reduces processing time. Only certain data fields are required to analyse the browsing behavior of computer users and Internet usage in the organisation. For example, the `client` and `userid` fields in the web proxy logs refer to the same person; therefore, only one field is required and the other is classified as irrelevant data. The following fields were deleted after the data cleaning process: `etime`, `client`, `log-tag` and `request object`.

Next, certain data transformations were performed on the web proxy logs. The `timestamp` of each request was converted into a more readable format, namely, 1121182139 was transformed to 2005/07/12 15:28 Thursday. Furthermore, strings (e.g., `day`, `request`, `url`, `userid` and `content type`) in the proxy logs were converted to numerical values to speed up processing.

4.2 Pattern Discovery

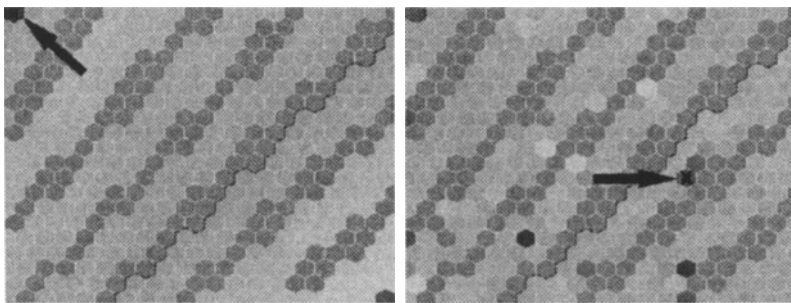
Pattern discovery draws on algorithms used in data mining, machine learning and pattern recognition to detect interesting patterns. These patterns can be further analysed during the pattern analysis phase to gain better insights into the data and to aid in evidence recovery. The SOM data mining technique was used for pattern discovery in our study.

Pattern discovery using a SOM occurs after the data pre-processing phase. Note that the SOM's learning process terminates when the SOM produces acceptable results or a pre-set limit is reached on the number of iterations.

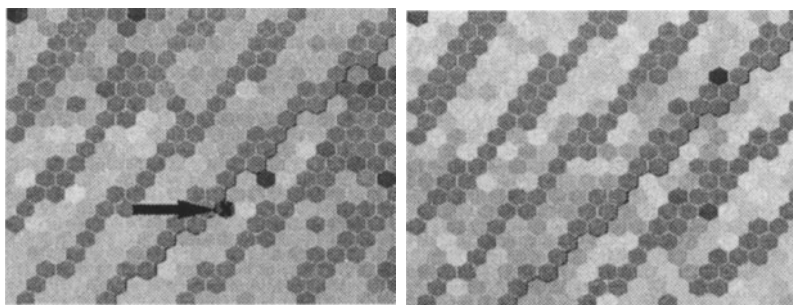
4.3 Pattern Analysis

Two-dimensional maps, which are displayed as hexagonal grids in Figure 1, are generated after the SOM completes its learning process. These "component maps" reveal variations in the values of components across the map. Each component map visualises the spread of values in a particular dimension. In the images in Figure 1, the color blue indicates low values, red indicates high values and the other colors (e.g., green and yellow) represent intermediate values. The color grey indicates that no data is mapped to that particular unit (or hexagonal grid). Color versions of the images are available at mo.co.za/bib.htm.

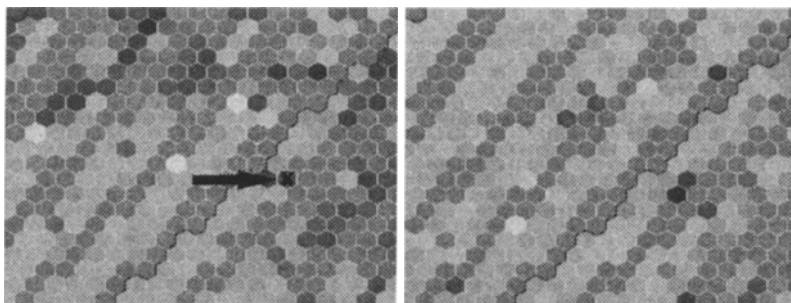
The two-dimensional component maps are a powerful visualisation aid. An application has been implemented to support the analysis of component maps by forensic investigators. Each unit in the map contains information about HTTP requests. A unit is selected by clicking on it,



(a) Component map (time); (b) Component map (day).



(c) Component map (request); (d) Component map (content type).

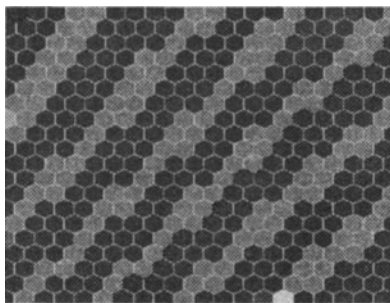


(e) Component map (URL); (f) Component map (userid).

Figure 1. Component maps generated from web proxy data.

(Color images are available at mo.co.za/bib.htm).

which causes it to be highlighted with a small checker board (see top left corner of Figure 1(b)). Information about the unit is then presented in a dialog box at the bottom of the screen (see Figure 2).



(g) Component map (size).

Figure 1 (continued). Component maps generated from web proxy data.
(Color images are available at mo.co.za/bib.htm).

4.4 Analysis of Component Maps

Figure 1(a) reveals variations in the time periods that computer users made HTTP requests. The map has three portions: blue (top left), green and red (bottom right). The blue portion denotes the 12 am to 9 am time period; green denotes 9 am to 3 pm, and red, 3 pm to 12 am. Upon viewing the map, it is immediately obvious that the green portion is significantly larger than the others, implying that Internet usage mostly occurred from 9 am to 3 pm.

Figure 1(b) presents variations in the specific days that HTTP requests were made. HTTP requests occurred mainly in the middle of the week: Tuesday, Wednesday and Thursday (green), as opposed to Friday, Saturday and Sunday (blue and red).

Figure 1(c) reveals variations in HTTP requests. POST (green) and GET (blue) were common requests. Usually, the GET method is by far the most common request for a specific URL; Figure 1(c) clearly shows that is, in fact, the case.

Figure 1(d) presents variations in the type of data returned by HTTP requests. The map shows that the content is predominantly images, applications and text (green and yellow), with text being the most common.

Figure 1(e) reveals variations in the URLs of the requests. The requests involved 4,263 distinct domain names, which were replaced with numerical values. Most of the requests involved domains that were mapped to lower numerical values (blue and green). The most popular domain was www.google.co.za (represented by 1), which was accessed 43,952 times (approximately 10% of requests).

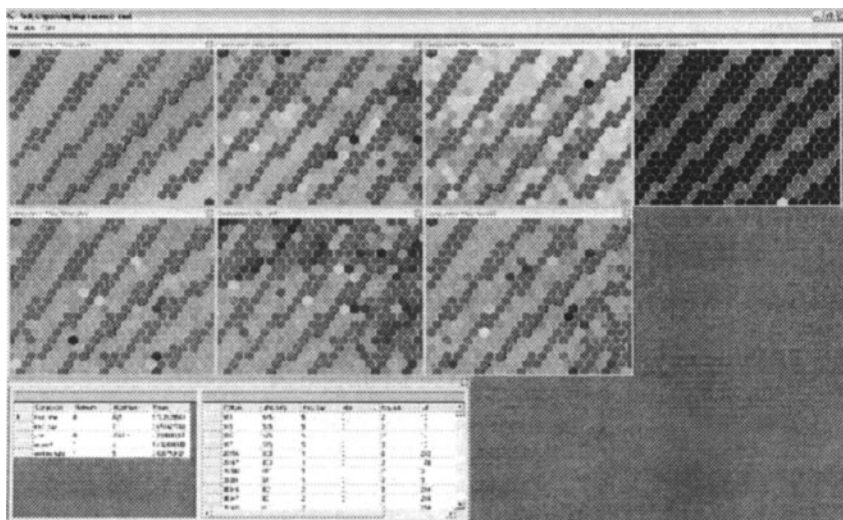


Figure 2. Screenshot of network forensics application.

Figure 1(f) presents variations with regard to users who made the HTTP requests. In particular, the map provides information about user browsing behavior and Internet usage.

Figure 1(g) reveals variations in the size of data returned by HTTP requests. At least 90% of the map is blue, corresponding to HTTP requests that returned data less than 1,000 KB in size. However, three units have a color other than blue; the corresponding HTTP requests were for data ranging from 1,000 KB to 7,000 KB.

Figure 2 presents a screenshot of an implemented application that presents all seven component maps to a forensic investigator. By comparing the maps, the investigator can review Internet usage and analyse the browsing behavior of users. The browsing behavior includes HTTP request times and URLs, the type of data requested from specific URLs, and the size of data requested from specific URLs,

Detecting anomalous browsing behavior is important in many network forensic investigations. Ideally, this is accomplished by examining the irregular portions of component maps—regions where a specific color has fewer occurrences. For example, in Figure 1(a), the irregular portion in the blue zone is marked with a cross (indicated with an arrow). It indicates HTTP requests made during an unusual time period (between 12 am and 6 am). However, an investigation of the URLs, and the type of data requested revealed no suspicious activity.

In Figure 1(b), the irregular portion is the red zone marked with a cross (indicated with an arrow), which corresponds to HTTP requests that were made on Saturday. The use of this map in conjunction with the other maps reveals possible correlations between the various dimensions. It appears a correlation exists between the red portions in Figures 1(b) and 1(e) (indicated with arrows). Upon investigating the URLs and the type of data requested, it was found that suspicious activities had in fact occurred. First, Figure 1(d) indicates that the majority of the requests were for images. Second, Figure 1(e) shows very few red regions, indicating that URLs represented by red were visited rarely. On examining the original proxy logs, it was observed that a particular user visited several adult web sites on a Saturday and the contents retrieved by the user were mainly images.

In Figure 1(c), the irregular portion of the map is again the red zone marked with a cross (indicated with an arrow), which corresponds to the CONNECT method. The CONNECT method is normally used to tunnel a connection through an HTTP proxy. By investigating the URLs and the type of data requested, it was found that a user was conducting Internet banking, which was not deemed to be an unauthorised activity.

Although certain activities deviate significantly from others, they may not necessarily be unauthorised or illegal. For example, in the example above, one anomalous incident involved Internet banking while the other involved visits to adult web sites. Therefore, when anomalous activity is suspected, it is necessary to conduct a detailed examination of the original web proxy logs.

5. Conclusions

Self-organising maps (SOMs) can be used in network forensics to analyse web proxy data with a view to investigating browsing patterns of computer users and detecting anomalous behavior. The component maps resulting from two-dimensional mappings of data produced by a SOM offer a powerful framework for visualising and analysing the large volumes in data contained in web proxy logs. By comparing different component maps, a forensic investigator can rapidly obtain an overview of Internet usage and an understanding of the browsing patterns of computer users, including anomalous behavior. Only when anomalous behavior is indicated, is it necessary for the investigator to conduct a detailed analysis of the web proxy logs. This can contribute to an increase in the quality of forensic investigations and a reduction in the amount of effort, especially in network forensics, which involves the collection and analysis of large quantities of data.

References

- [1] A. Abraham and V. Ramos, Web usage mining using artificial ant colony clustering and linear genetic programming, *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1384-1391, 2003.
- [2] B. Berendt, Web usage mining, site semantics and the support of navigation, *Proceedings of the Workshop on Web Mining for E-Commerce: Challenges and Opportunities*, 2000.
- [3] J. Brittle and C. Boldyreff, Self-organizing maps applied in visualising large software collections, *Proceedings of the Second International Workshop on Visualising Software for Understanding and Analysis*, 2003.
- [4] M. Caloyannides, *Privacy Protection and Computer Forensics*, Artech House, Boston, Massachusetts, 2004.
- [5] R. Cooley, B. Mobasher and J. Srivastava, Data preparation for mining World Wide Web browsing patterns, *Knowledge and Information Systems*, vol. 1(1), pp. 5-32, 1999.
- [6] V. Corey, C. Peterman, S. Shearin, M. Greenberg and J. van Bokkelen, Network forensics analysis, *IEEE Internet Computing*, vol. 6(6), pp. 60-66, 2002.
- [7] G. Deboeck, Financial applications of self-organising maps, *Neural Network World*, vol. 8(2), pp. 213-241, 1998.
- [8] M. Eirinaki and M. Vazirgiannis, Web mining for web personalization, *ACM Transactions on Internet Technology*, vol. 3(1), pp. 1-27, 2003.
- [9] A. Engelbrecht, *Computational Intelligence: An Introduction*, Wiley, Chichester, United Kingdom, 2002.
- [10] M. Géry and H. Haddad, Evaluation of web usage mining approaches for users' next request prediction, *Proceedings of the Fifth ACM International Workshop on Web Information and Data Management*, pp. 74-81, 2003.
- [11] T. Kohonen, The self-organizing map, *Proceedings of the IEEE*, vol. 78(9), pp. 1464-1480, 1990.
- [12] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin-Heidelberg, Germany, 2001.
- [13] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero and A. Saarela, Self organization of a massive document collection, *IEEE Transactions on Neural Networks*, vol. 11(3), pp. 574-585, 2000.

- [14] P. Kolar and A. Joshi, Web mining: Research and practice, *IEEE Computing in Science and Engineering*, vol. 6(4), pp. 49-53, 2004.
- [15] R. Kosala and H. Blockeel, Web mining research: A survey, *SIGKDD Explorations*, vol. 2(1), pp. 1-15, 2000.
- [16] Y. Li, X. Chen and B. Yang, Research on web-mining-based intelligent search engines, *Proceedings of the International Conference on Machine Learning and Cybernetics*, 2002.
- [17] C. Maltzahn and K. Richardson, Performance issues of enterprise-level web proxies, *Proceedings of the ACM Sigmetrics International Conference on Measurement and Modeling of Computer Systems*, pp. 13-23, 1997.
- [18] B. Mobasher, N. Jain, E. Han and J. Srivastava, Web Mining: Pattern Discovery from World Wide Web Transactions, Technical Report TR96-050, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, Minnesota, 1996.
- [19] S. Mukkamala and A. Sung, Identifying significant features for network forensic analysis using artificial techniques, *International Journal of Digital Evidence*, vol. 1(4), 2003.
- [20] M. Noblett, M. Pollitt and L. Presley, Recovering and examining computer forensic evidence, *Forensic Science Communications*, vol. 2(4), 2000.
- [21] U. Payer, P. Teufl and M. Lamberger, Traffic classification using self-organizing maps, *Proceedings of the Fifth International Networking Conference*, pp. 11-18, 2005.
- [22] M. Reith, C. Carr and G. Gunsch, An examination of digital forensic models, *International Journal of Digital Evidence*, vol. 1(3), 2002.
- [23] K. Smith and A. Ng, Web page clustering using a self-organizing map of user navigation patterns, *Decision Support Systems*, vol. 35(2), pp. 245-256, 2003.
- [24] J. Srivastava, R. Cooley, M. Deshpande and P. Tan, Web usage mining: Discovery and applications of usage patterns from web data, *SIGKDD Explorations*, vol. 1(2), pp. 12-23, 2000.
- [25] S. Tangsripairoj and M. Samadzadeh, Application of self-organizing maps to software repositories in reuse-based software development, *Proceedings of the International Conference on Software Engineering Research and Practice*, vol. 2, pp. 741-747, 2004.
- [26] J. Vacca, *Computer Forensics: Computer Crime Scene Investigation*, Charles River Media, Hingham, Massachusetts, 2002.
- [27] J. Vesanto, Using SOM in Data Mining, Licentiate Thesis, Helsinki University of Technology, Helsinki, Finland, 2000.

- [28] J. Vesanto, Data Exploration Process Based on the Self-Organizing Map, Doctoral Thesis, Helsinki University of Technology, Helsinki, Finland, 2002.
- [29] D. Wessels, Squid Web Proxy Cache (www.squid-cache.org).

Chapter 21

GSM CELL SITE FORENSICS

Christopher Swenson, Tyler Moore and Sujeet Shenoi

Abstract Cell site forensics is a new and growing area of digital forensics, enabling investigators to verify a mobile phone subscriber's location at specific times. This paper focuses on cell site forensics in GSM networks. In particular, it discusses current methods utilizing call detail records generated from telephone switches that provide information about cellular calls and text messages, and the cellular towers on which calls/messages were placed and received.

Keywords: GSM networks, cell site forensics, subscriber location estimation

1. Introduction

Cell phones are small, mobile, integrated communications and computing devices. They have become indispensable to the daily activities of much of the world's population. As such, cell phones are data repositories, holding evidence of legal—and illegal—activities. Specifically, digital evidence is stored in cell phone SIM cards, internal memory chips and external memory devices. Numerous techniques and tools have been developed for extracting and analyzing evidence from cell phones and peripherals.

Less well-known, but equally important for evidentiary purposes, is information about mobile subscribers and phone calls that is stored within the mobile communications network infrastructure. Mobile networks maintain information about who called whom, from where, when, and for how long [3]. This information can pinpoint a mobile subscriber's location at a specific time and the subscriber's movement over time. It can demonstrate the mobile subscriber's presence at the scene of a crime. Moreover, historical location-time data pertaining to mobile subscribers may provide details about the “dynamics” of a crime, from planning to execution.

Please use the following format when citing this chapter:

Swenson, C., Moore, T., Shenoi, S., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 259–272.

Cell site forensics involves the application of scientific techniques to analyze mobile communications network data. Current methods examine call detail records (CDRs) that are created by telephone switches for billing purposes. CDRs are generated, for example, whenever a subscriber makes or receives a call, sends or receives a text message, or moves to a new area of cell phone coverage. They provide detailed information about cellular calls and text messages (e.g., caller/sender, called party/receiver and time of call/message). CDRs identify the cellular towers on which calls were placed and received. These cell ids provide location information that can be refined using other data maintained by service providers, e.g., directions (azimuths) of mobile subscribers from cellular tower antennae and the power levels of subscriber to cellular tower communications.

This paper focuses on cell site forensics for GSM (Global System for Mobile Communications) networks [2, 5]. GSM is the largest mobile communications system in the world, and the fastest growing network in the United States [6]. The following sections discuss the basic concept of cell site forensics and highlight strategies for obtaining accurate time-location information from GSM networks.

2. GSM Cellular Networks

Cellular networks differ from traditional wireless telecommunications networks in three respects. First, small base stations or “cells” are used instead of large antennae. Each cell has a range of a few miles or less, which reduces the need for large, power-consuming equipment. Also, this provides greater overall bandwidth as a given frequency can be allocated to multiple cells. Second, the cell-based structure requires a mechanism for handovers—allowing users to seamlessly move between different coverage areas without dropping calls. Finally, network bandwidth released by subscribers can be reused by other subscribers.

This paper focuses on GSM networks [2, 5]. GSM was originally deployed in the early 1990s as an international standard for digital cellular networks. It is now the largest mobile communications system in the world [6]. GSM’s popularity partly stems from the fact that subscribers can use the same phone in multiple countries.

Figure 1 presents a schematic diagram of a GSM network, including its connections to the public switched telephone network (PSTN) and the Internet. The portions of the network concerned with mobile telecommunications are often referred to as the public land-mobile network (PLMN). The main components of a GSM network are described below.

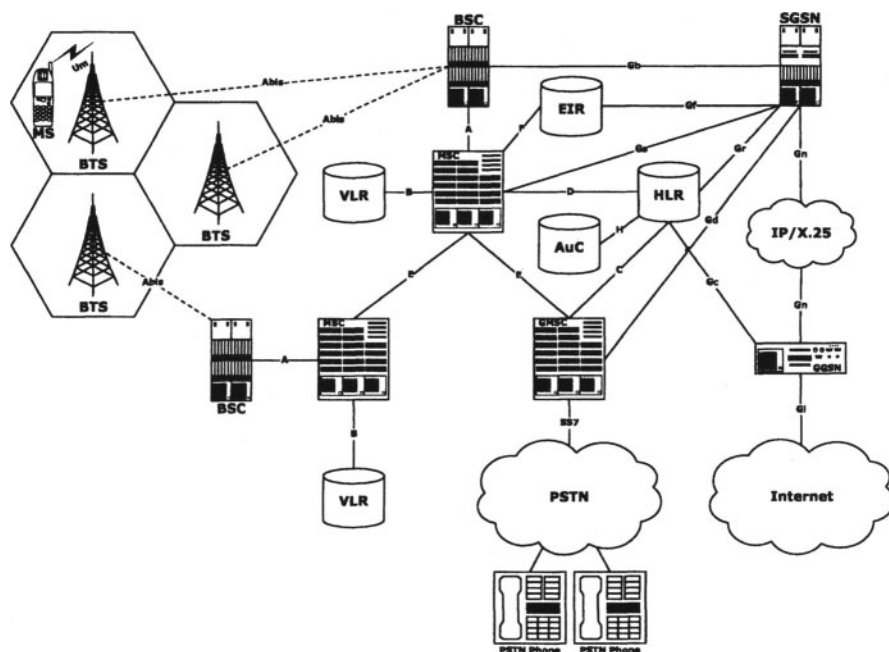


Figure 1. Schematic diagram of a GSM network.

Mobile Subscriber (MS): A mobile subscriber includes the functionality of the handset and the SIM card, which are identified by the International Mobile Equipment Identifier (IMEI) and International Mobile Subscriber Identifier (IMSI), respectively. An MS communicates with the network indirectly through cell towers known as Base Transceiver Stations (BTSs).

Base Transceiver Station (BTS): A BTS communicates with mobile subscribers using the Um interface and with the Base Station Controller (BSC) using the Abis interface. A BTS has limited decision-making duties and has practically no record storage capabilities.

Base Station Controller (BSC): A BSC is responsible for coordinating the communications of BTSs and MSCs, using the Abis and A interfaces, respectively. A BSC maintains information about the current locations of all mobile subscribers in its zone of control. However, very little of this information is stored for extended periods of time.

Mobile Switching Center (MSC): The MSC is the main decision-making entity in a GSM network. It relays communications within its child BSCs and with external MSCs using the A and E interfaces, respectively. Links between MSCs facilitate communications between mobile subscribers on different network segments and between subscribers with

different service providers. MSCs typically generate call detail records (CDRs) and collect billing information.

Gateway MSC (GMSC): The GMSC is a portal from a mobile network to other networks, especially the public switched telephone network (PSTN). A GMSC connects GSM networks to the PSTN using standard SS7 protocols, e.g., ISUP, TCAP, SCCP [1, 10], allowing calls placed on one network to be routed to another. A GMSC communicates with other MSCs using the E interface and it locates mobile subscribers by querying the Home Location Register (HLR) using the C interface.

Home Location Register (HLR): The HLR is a central repository for the current MSC locations of all subscribers. A PLMN typically has a small number of HLRs, each of which can serve several hundred thousand subscribers. An MSC uses the D interface to query an HLR for the purposes of authenticating and locating users. A GMSC communicates with an HLR using the C interface to locate users, and the Gr and Gc interfaces for GPRS location and authentication.

Visitor Location Register (VLR): The VLR is similar to an HLR, except that it contains information about mobile subscribers on a particular MSC. An MSC uses the B interface to query a VLR to determine if a subscriber is located on the MSC. The load on the master HLR is decreased because the MSC interrogates the VLR before querying the HLR.

Authentication Center (AuC): The AuC is responsible for verifying a mobile subscriber's identity and for generating keying information using the A3 and A8 cryptographic procedures. When implementing the subscriber authentication procedure, the HLR caches the A3/A8 outputs from the H interface and forwards them to the MSC.

Equipment Identity Register (EIR): The EIR maintains and enforces a blacklist of mobile handsets, mainly to curb fraud and theft. The EIR communicates with an MSC using the F interface and with a Serving GPRS Support Node (SGSN) using the Gf interface.

Serving GPRS Support Node (SGSN): The SGSN connects mobile subscribers to advanced data services provided by General Packet Radio Service (GPRS) [8]. The SGSN coordinates data sessions with mobile subscribers and relevant GGSNs using the Gn interface and the GPRS IP/X.25 core.

Gateway GPRS Support Node (GGSN): A GGSN serves as an access point for SGSNs to access a desired service, e.g., WWW and POP3. A GGSN usually communicates with SGSNs using the Gn interface and with external networks using the Gi interface.

3. Call Control Procedures

This section describes call control procedures for phone calls involving PSTN and GSM subscribers. Details of call setup and teardown procedures are provided for: (i) PSTN to PSTN calls, (ii) PSTN to GSM calls, (iii) GSM to PSTN calls, and (iv) GSM to GSM calls. These procedures provide insight on the information requirements and strategies involved in cell site forensics. Note that PSTN to PSTN calls *per se* are not relevant to GSM network forensics as the calls remain within the PSTN. However, they are discussed because PSTN call setup and teardown procedures are involved in PSTN to GSM and GSM to PSTN calls.

Call setup and teardown in GSM networks build on call control procedures used in ISUP, the basic call handling protocol for PSTN (SS7) calls. However, GSM call control requires extra steps to allocate radio frequencies in addition to voice trunks. Also, it incorporates cellular authentication and encryption mechanisms. Readers are referred to [1, 10] and [4] for details about ISUP and GSM call control procedures, respectively, and [9] for details about forensic techniques involving the ISUP protocol in SS7 networks.

3.1 PSTN to PSTN Calls

PSTN to PSTN call setup follows a four-step procedure involving the exchange of ISUP protocol messages.

1. The PSTN caller dials the receiver's phone number, which is interpreted by the caller's central office switch.
2. Initial address messages (IAMs) are sent from the caller's central office switch to the receiver's central office switch. The IAMs attempt to reserve a chain of voice trunks between the caller's and receiver's switches.
3. Address complete messages (ACMs) are sent back along the same path from the receiver's switch to the caller's switch, signifying that voice trunks have been successfully allocated and the receiver's phone is ringing. If the receiver is not available, the switch returns a release (REL) message with the appropriate cause code set (e.g., user busy or no answer).
4. An answer message (ANM) is sent from the receiver's switch to the caller's switch, indicating that the receiver has picked up the phone. Billing is then initiated. If the call is not completed, the call attempt will most likely not be registered in billing records,

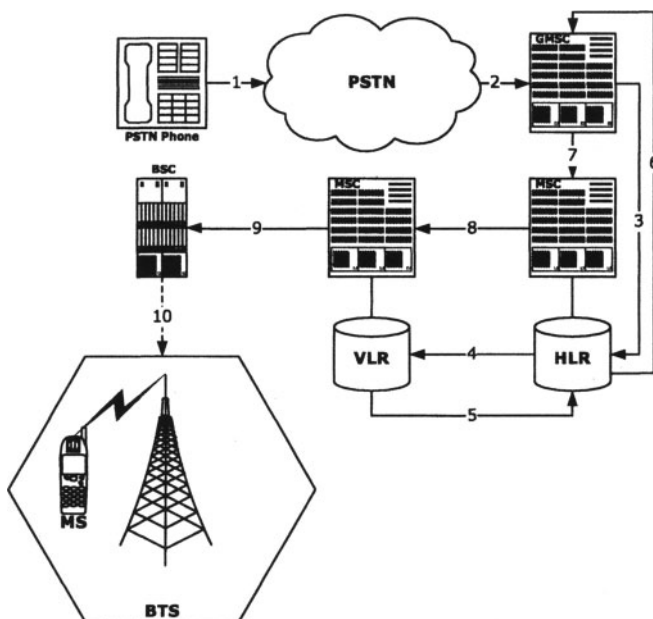


Figure 2. Call setup procedure for a PSTN to GSM call.

although it may be registered in call detail records (CDRs) generated at the switch.

PSTN calls are terminated by a sequence of release (REL) and release complete (RLC) messages. If a call is terminated within a few seconds of its initiation, the call is usually not charged and no billing record may be generated.

3.2 PSTN to GSM Calls

Call setup in GSM networks involves the exchange of numerous messages between various network components. To simplify the presentation, we list only the basic steps involved in setting up calls from PSTN to GSM networks (Figure 2).

1. The PSTN caller dials the mobile subscriber's telephone number, which is interpreted by the PSTN switch.
2. The PSTN switch generates the relevant ISUP IAM and routes it to the GMSC belonging to the service provider of the mobile subscriber.

3. The GMSC queries the HLR to identify the mobile subscriber roaming number (MSRN) for the receiver's phone number. The MSRN is the phone number assigned to the mobile phone on the network where it is currently located; it helps the switch route the call correctly. For example, suppose the receiver's cell phone number is (918) 555-1234, which has the 918 area code corresponding to Tulsa, Oklahoma. If the receiver is currently located in Oklahoma City, Oklahoma, he would be assigned an MSRN with the 405 Oklahoma City area code, e.g., (405) 555-6789. Thus, the switch would know to route the call to the receiver in Oklahoma City instead of Tulsa.
4. The HLR queries the last known VLR where the receiver was located to check if the user is still at that location and is connected to the network.
5. The VLR replies to the HLR's query with the receiver's status.
6. The HLR replies to the GMSC's query, identifying the MSC for routing the call.
- 7,8. The GMSC forwards the IAM through the GSM network core.
9. The MSC initiates call control procedures for setting up the voice call to the mobile subscriber (receiver). It notifies the BSC that the subscriber is receiving the call.
10. The BSC notifies the mobile subscriber (MS) via the BTS of the incoming call, and begins to check the receiver's credentials and allocate resources, e.g., voice circuits and radio frequencies.

Billing is initiated as soon as the mobile subscriber (receiver) answers the call, and the call is noted in the MSC. A call terminated within a few seconds of initiation is usually not billed and no CDR may be generated.

Call teardown can be initiated in many ways, e.g., when a subscriber goes out of range, hangs up, or runs out of minutes. When the MSC notes that one of the parties ends the call, it begins to teardown the call for the mobile subscriber by releasing its resources, while notifying the remote party that the call is complete by performing the ISUP teardown procedure (a REL message followed by a RLC message are sent to both parties). Procedures for tearing down GSM to PSTN calls and GSM to GSM calls are practically identical.

3.3 GSM to PSTN Calls

The call setup procedure for a GSM to PSTN call is relatively similar to the setup procedure for a PSTN to GSM call. Some clarifications are provided below.

1. After determining that the call is not local via VLR and HLR queries, an IAM is sent to the GMSC to route the call to the PSTN.
2. While waiting for the ACM from the GMSC, the MSC initiates the call setup procedure for the GSM subscriber (caller), allocating the required radio frequencies and performing authentication.
3. Upon receiving the ANM from the GSMC, the call is routed from the GMSC to the subscriber (caller).

3.4 GSM to GSM Calls

Call setup procedures for GSM to GSM calls can vary significantly according to the circumstances of the two subscribers. If the two mobile phones have different service providers or are located in different countries, the call setup procedure is similar to that for the PSTN-initiated and GSM-initiated calls described above.

If the GSM subscribers are on the same network, the MSC first queries the VLR to determine if the call is local (in which case it does not have to reserve voice trunks outside the MSC). If the call is local, radio allocations are made to both subscribers and voice communications are routed through the MSC. If the subscriber (receiver) is not local, the MSC queries the HLR to determine where to route the call. An IAM is then sent to the appropriate MSC, and the call proceeds in the same way as PSTN-initiated and GSM-initiated calls.

4. Call Records

This section describes the data pertaining to cell phone calls that are generated within the network and often stored by service providers. The data are indispensable to cell site forensics, especially in obtaining historical time-location information about mobile subscribers.

As described in the previous section, signaling messages are generated whenever a call is initiated in a GSM network. The switches at both ends of the call may record details about calls if certain conditions are satisfied. For example, a switch may be configured to generate records for calls only if they complete, for calls that are more than a few seconds

long, or for all calls that are attempted. In general, record generation and record storage times vary from provider to provider.

4.1 Call Detail Records

Call detail records (CDRs) are generated and stored by telephone network switches. CDRs are consolidated at fixed intervals to create billing records, which only contain information related to customer billing. As such, billing records tend to be fewer in number than CDRs and contain less information about network events.

GSM 12.05 [3] originally specified seventeen types of CDRs for logging events. Almost all the CDRs indicate mobile subscriber location and time. However, many of the CDRs are optional and others may only be partially recorded.

The most commonly generated CDRs are: (i) Mobile originated/terminated call records, (ii) Mobile originated/terminated SMS records, (iii) HLR location update records, and (iv) VLR location update records. The following subsections provide details about these records, which are particularly useful for cell site forensics. The records are only partially specified for reasons of space. Important fields are displayed along with their descriptions and whether they are mandatory (M), conditional (C) or optional (O).

Mobile Originated/Terminated Call Records: Mobile originated call (MOC) records are typically generated by the originating MSC for all outgoing call attempts. Mobile terminated call (MTC) records are generated upon call termination. MOC and MTC records may be generated even when call attempts fail. Table 1 presents the MOC record fields; MTC records have almost identical fields. Note that for a mobile originated call, only the origination cell id is recorded. On the other hand, for a mobile terminated call, only the destination cell id is recorded. In general, an MSC has limited information about a call on the other end, and it sees little difference between PSTN originated and mobile originated calls.

Mobile Originated/Terminated SMS Records: SMS mobile originated (SMS-MO) records are typically produced by the originating MSC for all SMS messages. Similarly, SMS mobile terminated (SMS-MT) records are produced by the terminating MSC for all SMS messages sent to subscribers in its jurisdiction. Relevant fields in SMS-MO records are shown in Table 2. The SMS-MT record structure is nearly identical, except that it has information about the receiver instead of the sender.

Table 1. Relevant mobile originated call record fields.

Field	Type	Description
Record Type	M	Mobile originated
Served IMSI	M	IMSI of calling party
Served IMEI	C	IMEI of calling party (if available)
Served MSISDN	O	Primary MSISDN of calling party
Called Number	M	Number dialed by caller
Translated Number	O	Called number after MSC translation
Connected Number	O	Actual connected number (if different)
Location	M	Cell id of originating call
Change of Location	O	Timestamped changes in location area and cell id
Event Timestamps	C	Incoming traffic channel assignment
	C	Answer
	O	Release
Call Duration	M	Duration of call or holding time

Table 2. Relevant mobile originated SMS record fields.

Field	Type	Description
Record Type	M	Mobile originated
Served IMSI	M	IMSI of sending party
Served IMEI	O	IMEI of sending party (if available)
Served MSISDN	O	Primary MSISDN of sending party
Recording Entity	M	Visited MSC identifier
Location	O	Location area code and cell id of origination
Event Timestamp	M	Time SMS received by MSC
SMS Result	C	Result of attempted delivery (if unsuccessful)

HLR Location Update Records: HLRs generate records about location updates and registrations as these are often billable events, e.g., when subscribers use their cell phones while traveling outside their home area. Table 3 presents the relevant fields of an HLR location update record.

VLR Location Update Records: VLRs keep track of updates and registrations of mobile subscribers. However, VLRs do not necessarily record information about subscribers leaving their jurisdictions for new VLRs. This is because it is the new VLR's responsibility to keep such records. The key fields in VLR location update records are shown in Table 4.

Table 3. Relevant HLR location update record fields.

Field	Type	Description
Served IMSI	M	IMSI of served subscriber
Recording Entity	M	HLR identifier
Old Location	O	VMSC identifier VLR identifier
New Location	M	VMSC identifier VLR identifier
Update Timestamp	M	Time update was invoked
Update Result	C	Result of location update (if unsuccessful)

Table 4. Relevant VLR location update record fields.

Field	Type	Description
Served IMSI	M	IMSI of served subscriber
Served MSISDN	O	Primary MSISDN of served subscriber
Recording Entity	M	Recording entity (MSC/VLR) identifier
Old Location		(Not present for registration)
	C	VMSC number
	C	Location area code
New Location	M	VMSC identifier
	M	Location area code
	O	Cell id
Update Timestamp	M	Time update was invoked
Update Result	C	Result of location update (if unsuccessful)

5. Tracking Mobile Subscribers

This section discusses how location information pertaining to mobile subscribers may be estimated. It also highlights techniques for refining and verifying location estimates.

5.1 Estimating Mobile Subscriber Locations

Because of their convenience, cell phones have become indispensable to coordinating and executing criminal activities. Cell phones have traditionally been viewed as more anonymous than normal landline phones, but this is not true. CDRs generated during mobile communications provide valuable information about the identities, caller groups and lo-

cations of criminal elements. Much of this information is preserved by service providers because it is relevant to billing, and is, therefore, available to criminal investigators.

CDRs for mobile calls (Table 1) and SMS messages (Table 2) contain IMSIs that identify subscribers, IMEIs that identify handsets, along with timestamped location information. Mobile call CDRs for a network contain the cell ids of all mobile terminated and mobile originated calls in its jurisdiction (Table 1). SMS message CDRs can record the cell ids for all SMS messages (Table 2). Depending on the network topology, the radius served by a cell tower with a specific cell id may range from a few hundred yards to several miles.

When a subscriber moves between different jurisdictions in a mobile communications network, HLR and VLR updates may be required to ensure that the subscriber continues to receive service. Thus, the CDRs generated by HLR and VLR location updates (Tables 3 and 4) contain timestamped information about the movement of the subscribers. Although cell ids are not necessarily recorded in the CDRs for HLR and VLR location updates, it is still possible to analyze the timestamped "roaming" records to determine subscriber locations, e.g., on a roadway between two cities.

5.2 Refining Location Estimates

CDRs provide useful information about subscriber locations, but their estimates may lack specificity and accuracy. Service providers may not create CDRs for all call/message events, the CDRs themselves may not have values for all their fields, and the CDRs may not be stored after their information is summarized to produce billing records. Furthermore, the cell ids listed in CDRs may not provide location information of sufficient accuracy. As mentioned above, the radius served by a cell tower may range from a few hundred yards to several miles. Of course, if it is known that a mobile subscriber is traveling in an automobile or train, it is possible to superimpose cell tower locations on a roadmap to obtain more accurate location estimates (possibly 20 to 100 feet).

Cellular network components often generate other useful information that can be used to refine subscriber location estimates. One example is the azimuth of a subscriber relative to a cell tower. A tower with a common triangular antenna has three possible azimuth values, corresponding to 0°, 120° and 240°. The specific angular direction of a mobile subscriber helps refine the location estimate within a cell site. The power level of a subscriber's communications with a cell tower can also help refine location estimates. Mathematical equations relating the power

level to the distance from a cell tower can be used to obtain location estimates within 230 feet [7].

Information for refining location estimates, such as azimuth values and power levels, can be obtained by analyzing a “cell site dump.” However, this task is arduous, time-consuming and expensive; typically, such analyses are reserved only for investigations of serious crimes. Note that this information is related to network management, not billing, and no standards govern its collection and storage. Consequently, not every service provider collects such information, and the information, even if collected, may not be stored for more than a few weeks.

5.3 Verifying Location Estimates

In cell site forensics it is extremely important to verify that the location estimates are indeed correct. This is typically done by visiting the probable locations at each cell site with the subscriber’s cell phone (if it has been seized). Alternatively, an identical model with the same service plan may be used; this information can be obtained using the IMEI and IMSI values in CDRs. The purpose of the test is to verify that had the cell phone been operating at the location at the time in question, it would have communicated with specific cell towers with the appropriate azimuth values and power levels. On-site testing helps determine whether or not certain cell towers were blocked from receiving mobile communications, for example, by buildings or landscape features. Since network topology and city topography can change fairly quickly, it is important to conduct the on-site verification of location estimates as soon as possible.

6. Conclusions

Cell site forensics provides information about a mobile subscriber’s location at specific times. It can place a suspect at or near a crime scene when the crime occurred, or it can provide exculpatory evidence that the suspect was present at some other location at that time. Cell site forensics also yields valuable historical location-time data from CDRs pertaining to a group of mobile subscribers (called a community of interest). This historical data can offer insights into the dynamics of a crime, from planning to execution.

But cell site forensics is not a panacea. It only identifies a “subscriber” (via the IMSI) and the handset (via the IMEI)—not the actual individual who used the cell phone at the time in question. Other types of evidence, e.g., eyewitness reports, closed circuit TV footage and DNA evidence, are required for purposes of attribution. However, cell site forensics,

because it provides location-time data, can help law enforcement agents identify the locations where this additional evidence might exist.

References

- [1] L. Dryburgh and J. Hewitt, *Signaling System No. 7 (SS7/C7): Protocol, Architecture and Services*, Cisco Press, Indianapolis, Indiana, 2005.
- [2] J. Eberspächer, H. Vögel and C. Bettstetter, *GSM: Switching, Services and Protocols*, John Wiley, Chichester, United Kingdom, 2001.
- [3] European Telecommunications Standards Institute, ETSI TS 100 616 V7.0.1 (1999-07), Digital Cellular Telecommunications System (Phase 2+), Event and Call Data (GSM 12.05 Version 7.0.1 Release 1998), 1998.
- [4] European Telecommunications Standards Institute, 3GPP TS 23.018 V7.1.0 (2005-09), 3rd Generation Partnership Project, Technical Specification Group Core Network and Terminals, Basic Call Handling, Technical Realization (Release 7), 2005.
- [5] A. Fares, *GSM Systems Engineering and Network Management*, 1stBooks Library, Bloomington, Indiana, 2003.
- [6] GSM World, GSM subscriber statistics (Q1 2005) (www.gsmworld.com/new/statistics), 2005.
- [7] M. Hellebrandt and R. Mathar, Location tracking of mobiles in cellular radio networks, *IEEE Transactions on Vehicular Technology*, vol. 48(5), pp. 1558-1562, 1999.
- [8] J. Hoffman, *GPRS Demystified*, McGraw-Hill, New York, 2003.
- [9] T. Moore, A. Meehan, G. Manes and S. Shenoi, Forensic analysis of telecom networks, in *Advances in Digital Forensics*, M. Pollitt and S. Shenoi (Eds.), Springer, New York, pp. 177-188, 2005.
- [10] T. Russell, *Signaling System #7*, McGraw-Hill, New York, 2002.

Chapter 22

AN ARCHITECTURE FOR SCADA NETWORK FORENSICS

T. Kilpatrick, J. Gonzalez, R. Chandia, M. Papa and S. Sheno

Abstract Supervisory control and data acquisition (SCADA) systems are widely used in industrial control and automation. Modern SCADA protocols often employ TCP/IP to transport sensor data and control signals. Meanwhile, corporate IT infrastructures are interconnecting with previously isolated SCADA networks. The use of TCP/IP as a carrier protocol and the interconnection of IT and SCADA networks raise serious security issues. This paper describes an architecture for SCADA network forensics. In addition to supporting forensic investigations of SCADA network incidents, the architecture incorporates mechanisms for monitoring process behavior, analyzing trends and optimizing plant performance.

Keywords: Process control systems, SCADA networks, network forensics

1. Introduction

Process control systems (PCSs) – often referred to as supervisory control and data acquisition (SCADA) systems – are widely used in manufacturing processes, chemical plant and refinery operations, and even for automation and climate control in office buildings [7]. In a typical SCADA implementation, sensors acquire data pertaining to process behavior; this data is passed to control algorithms implemented in the SCADA system. Depending on the sensor data and control objectives, output signals are sent to actuators that adjust process inputs and move the process to the desired state.

In many industrial environments, sensors, actuators and control software are deployed in different locations, which requires the implementation of a communications infrastructure. Consequently, modern SCADA protocols, e.g., Modbus [16, 17] and DNP3 [21, 22], now include specifi-

Please use the following format when citing this chapter:

Kilpatrick, T., Gonzalez, J., Chandia, R., Papa, M., Sheno, S., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II. eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 273–285.

cations for transporting sensor data and control signals using TCP/IP. TCP/IP is becoming the dominant transport mechanism in SCADA networks. Also, TCP/IP is facilitating interconnections between previously isolated SCADA networks and corporate IT infrastructures via gateway devices [6, 11].

The use of TCP/IP as a carrier protocol for SCADA systems and the interconnection of SCADA and IT networks raises serious security issues [9, 10, 13, 23]. Because SCADA systems were historically isolated from other networks, most SCADA protocols were designed without any security mechanisms. Transporting a SCADA protocol over TCP/IP means that an attack on the TCP/IP carrier can severely expose the unprotected SCADA protocol. Furthermore, the connectivity between IT and SCADA networks means that successful attacks on an IT network and its gateway devices could tunnel into a SCADA network, wreaking havoc on the industrial process [10].

This paper describes an architecture for SCADA network forensics. In addition to supporting forensic investigations of SCADA network incidents, the architecture incorporates mechanisms for monitoring process behavior, analyzing trends and optimizing plant performance.

2. SCADA Network Overview

SCADA networks have myriad variants ranging from small, locally centralized networks such as those used in simple manufacturing plants to vast, distributed networks used in refineries, oil and gas pipelines, and power distribution facilities. Figure 1 provides a generic view of a SCADA network, which is used throughout this paper as a framework for discussing the functional, design and interconnectivity elements of SCADA networks.

A SCADA network has two main components: the control center (top left corner of Figure 1) and the plant it controls (Site A in Figure 1). The two components are connected to each other via a SCADA server.

The control center is the hub of SCADA network operations. Its components include human machine interfaces (HMIs), engineering workstations, plant data historians, databases and various shared resources. Control center components communicate with each other using the management network, and with the plant (Site A) and other SCADA networks using the SCADA server.

The SCADA server functions as the sole interface between the control center and one or more sites for which the control center is responsible. The server is usually implemented with vendor specific software and its services are often based on the OPC standard.

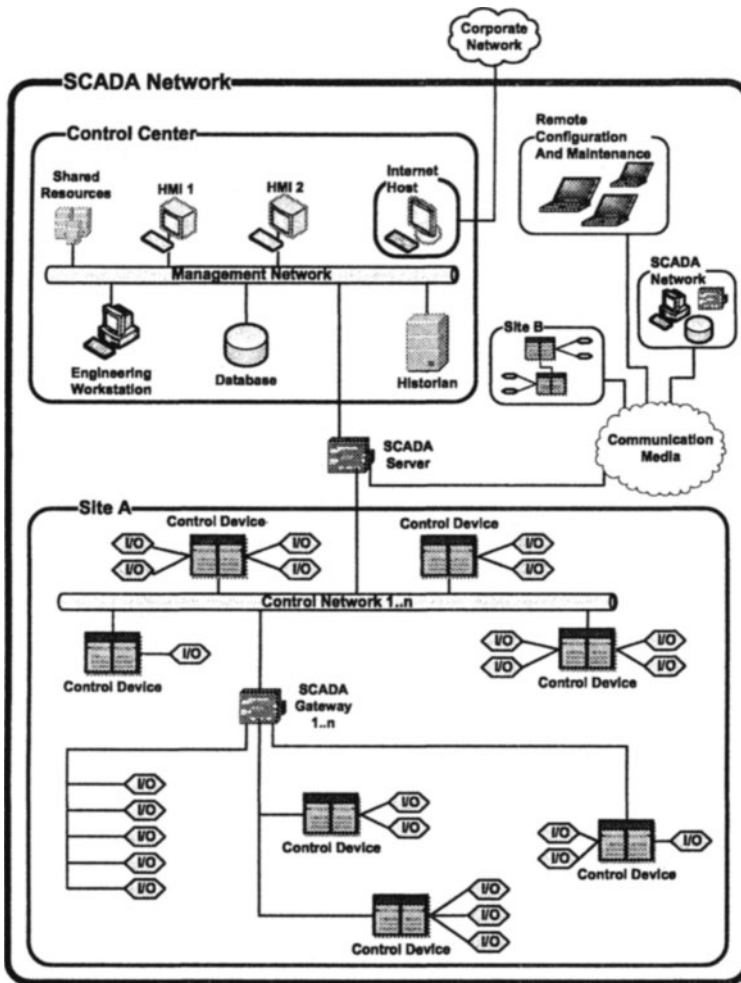


Figure 1. Generic SCADA network architecture.

The site or plant is the actual process system. From the process control point of view, it has three types of components: control devices, I/O devices (sensors and actuators) and the SCADA gateway. A control network interconnects components within a site. The 1..*n* notation is used in Figure 1 to express the fact that control network components may employ different protocols, e.g., Modbus and DNP3.

Control devices, which implement the process control logic, interface with and manipulate I/O devices. They include programmable logic controllers (PLCs), remote terminal units (RTUs), input/output controllers (IOCs) and intelligent electronic devices (IEDs). I/O devices

include sensors and actuators. Sensors measure specific process parameters (e.g., temperature and pressure) while actuators perform control actions (e.g., opening or closing a valve) to effect the desired changes to the process parameters.

The SCADA gateway is conceptualized at a high level of abstraction as a device that interfaces all control network components that cannot communicate directly with the SCADA server. Depending on its functionality, any control device can serve as a SCADA gateway. Special units called front-end processors (FEPs) are commonly used as SCADA gateways in industrial settings.

As mentioned earlier, control center components include human machine interfaces (HMIs), engineering workstations, databases, plant data historians and various shared resources. HMIs permit human operators in the control center to interact with process systems at a site in a limited manner. An operator at an engineering workstation has more authority over the SCADA network, and can reconfigure control devices and modify control algorithms (e.g., ladder logic) as needed.

Relevant data pertaining to process parameters and control actions is recorded in a database. Engineering workstation and HMI operators interact with the database to access and modify process data and control variables.

Historians archive data about SCADA network activities. The data includes sensor data, control actions effected by engineering workstation and HMI operators, and management network logs.

The management network may also contain various shared resources (e.g., printers, fax machines and file servers), but these are typically not considered part of the SCADA network. The generic SCADA network architecture in Figure 1 incorporates a separate Internet host that is connected to the corporate network. No physical connection exists between the SCADA management network and the Internet host. However, it is increasingly common for corporate networks to interconnect with previously isolated SCADA networks.

3. SCADA Network Forensics

The industrial use of SCADA systems has traditionally emphasized human and plant safety followed by the continuity of operations (availability). These requirements were maintained by isolating SCADA systems. However, the relatively recent encroachment of corporate intranets and even the open Internet into plant environments has made SCADA security a major issue. SCADA security, which was emphasized in Presidential Decision Directive 63 [24], has manifested itself in numerous

guidelines and standards promulgated by industry and government organizations such as AGA [1, 2], ANSI/ISA [3, 4], API [5] and NIST [18]. These guidelines and standards have been embraced by industry, and strong efforts are being undertaken across all sectors to secure SCADA systems [8, 14].

Forensics becomes relevant when security is breached [15]. SCADA network forensics involves the detailed examination of evidence related to security incidents. The goal is to discover the *modus operandi* and the identity of the perpetrator, as well as what went wrong so that the system can be hardened to prevent future incidents.

An architecture that supports SCADA network forensics can enhance industrial operations. Network forensics cannot be performed without mechanisms that systematically capture relevant traffic and state information throughout the network [15, 19, 20]. In the context of SCADA networks, the capture and subsequent analysis of sensor data and control actions assists in monitoring process behavior, examining trends and ultimately optimizing plant performance.

SCADA networks differ from corporate IT networks in several respects.

- **Security Principles:** Availability is the primary concern in most SCADA networks, followed by integrity and confidentiality. Corporate networks often emphasize confidentiality.
- **Protocol Isolation:** SCADA protocols, even those employing a TCP/IP carrier, are not used in an IT network. However, SCADA networks use traditional network protocols and are, therefore, vulnerable to attacks originating from or targeting IT networks.
- **Protocol Variations:** Most SCADA protocols are based on open standards. However, vendors often augment protocols or employ proprietary out-of-band mechanisms to provide additional functionality. On the other hand, protocol standards are enforced more strictly in traditional IT networks to support interconnectivity and interoperability.
- **Traffic Uniformity:** IT networks mainly have user-generated traffic. Consequently, the communication patterns are difficult to predict. SCADA network traffic is routine and predictable.
- **Traffic Volume:** IT networks have huge traffic volumes. SCADA networks have very light traffic. The messages tend to be shorter (e.g., Modbus messages never exceed a few hundred bytes) and fewer messages are transmitted.

- **Network Forensics:** Forensics in large-scale IT networks is extremely complicated and expensive [19, 20]. On the other hand, SCADA network forensics may be considerably simpler. Traffic uniformity and lower traffic volume in SCADA networks makes it possible to log relevant process/control data associated with every message and to subsequently analyze the data.

4. Incorporating Forensic Capabilities

This section describes an architecture that supports *post mortem* analysis of SCADA network traffic. The architecture employs “forensic agents” at strategic locations within a SCADA network. These agents forward relevant portions of network packets to a central location for storage and subsequent retrieval. A complete SCADA network history is obtained by reconstructing network events from packet fragments captured from locations throughout the SCADA network.

The architecture presented in Figure 2 incorporates two main entities: agents and a data warehouse. An agent captures SCADA network traffic in its local network segment and forwards a synopsis [19, 20] of each packet to the data warehouse. The data warehouse analyzes each packet synopsis and creates a data signature that it stores along with the synopsis in a storage area designated for the appropriate agent. The data warehouse also supports queries on the stored data. Note that an isolated network is used for all communications between agents and the data warehouse.

Generally, a SCADA network will have several types of agents and one data warehouse. A Level 1 agent is connected directly to the management network. Level 2 agents are located directly on the control networks. Level 3 agents are positioned downstream from the SCADA gateway. Note that the positioning of Level 3 agents is harder to categorize as the structures of SCADA networks differ considerably based on the industrial processes being controlled.

The data warehouse is typically housed in a secure location within the control center, but is not connected to the management network. Thus, the data warehouse is in close proximity to human operators, but is not vulnerable to exploits on the network that it monitors.

Some large industrial systems incorporate multiple interconnected SCADA networks. In such architectures, it is necessary to log traffic across different SCADA networks. This is accomplished by placing agents between the SCADA networks (Figure 3). The agents forward data to a common data warehouse that is not associated with a particular SCADA network. These units are referred to as Level 0 agents and

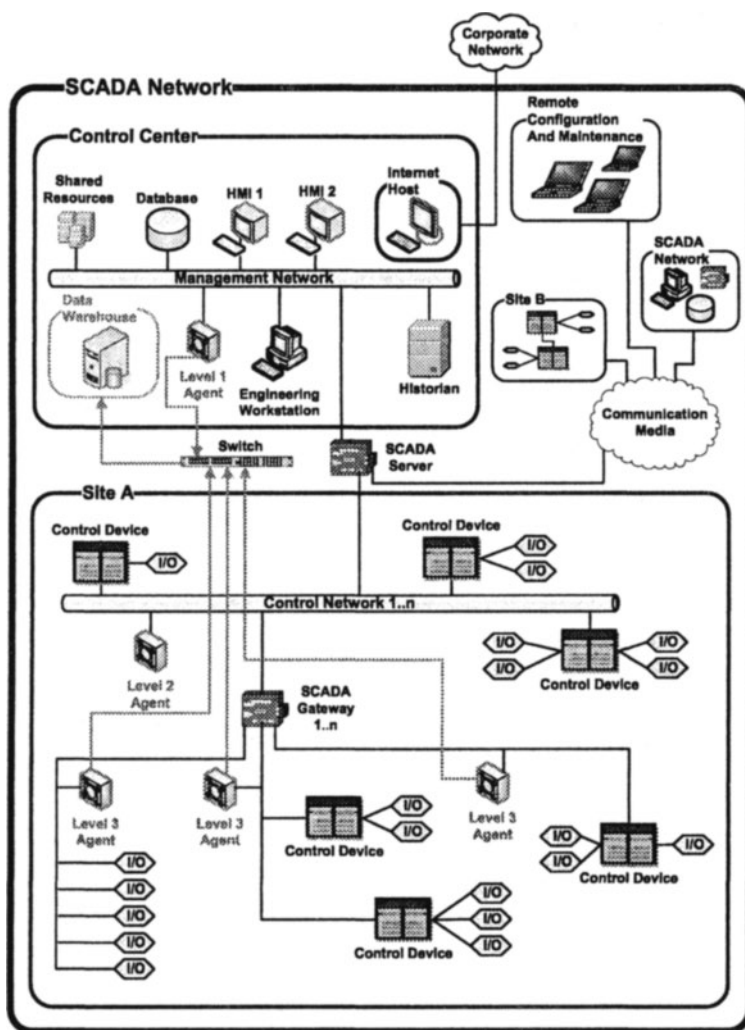


Figure 2. Generic SCADA network architecture with forensic capabilities.

Level 0 data warehouses. Note that depending on the interconnected network topology, there may be one or more Level 0 agents. We will see later that, in order to facilitate robust querying, the Level 0 data warehouse must be connected via a closed network to the data warehouses of the individual SCADA networks.

5. SCADA Traffic Collection and Analysis

Forensic agents capture and analyze SCADA network traffic. They create synopses of network packets that contain information relevant

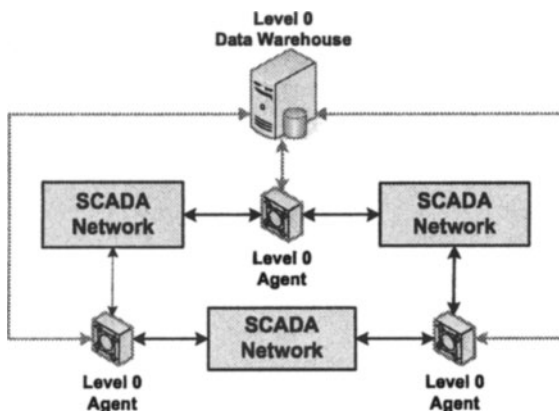


Figure 3. Communication between SCADA networks.

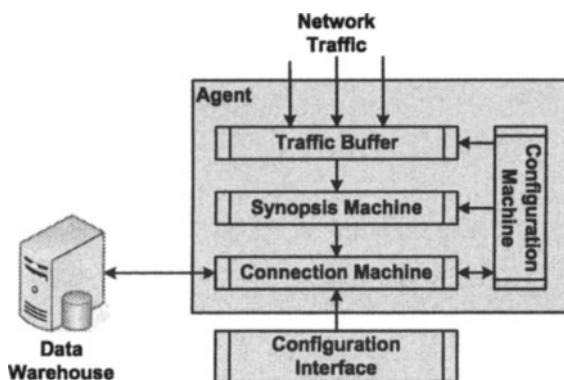


Figure 4. SCADA packet synopsis generation.

to forensic analysis [19, 20]. An agent incorporates a network traffic buffer, synopsis machine, connection machine and configuration machine (Figure 4).

The traffic buffer stores unprocessed network traffic. It employs a multi-threaded implementation of the standard producer/consumer algorithm and a bounded buffer. The `libpcap` [12] interface provides a mechanism for capturing raw Ethernet packets. Currently, development is focused on SCADA protocols (e.g., Modbus and DNP3) that can be transported using Ethernet frames. Future development plans also include support for serial communications (e.g., RS 232/485).

The synopsis machine constitutes the core of a forensic agent. It examines each packet from the traffic buffer and generates a packet synopsis

in the manner specified by the configuration rules. Note that partial synopses are produced for each encapsulating protocol. For example, for the OSI model, agents could produce layer 3 (network) and layer 4 (transport) synopses. Partial synopses are combined with location information and timestamps to produce a synopsis that is stored.

The connection machine supports agent communication. Secure communication is achieved by requiring architectural components to register with an authentication engine and using access control lists (ACLs) to implement mutual authentication.

Agent configuration is critical to the success of the forensic architecture. The configuration machine provides a mechanism to regulate operation of the agent at various levels (Figure 4), e.g., security settings, device registration, synopsis settings and protocol recognition modules. External devices attempting to configure an agent must be registered with the authentication engine and must use a common configuration interface. Some security settings are similar to those employed in IT networks, while others, such as synopsis settings, are unique to this architecture.

Proper configuration of the synopsis engine is important because of its role in the architecture. Two methods may be employed: level-based configuration and manual configuration. The level-based method configures agents according to their location, allowing agents to be configured with pre-defined synopsis algorithms. Agents are then configured as Level 0 (between SCADA networks), Level 1 (management network), Level 2 (control network) or Level 3 (behind a SCADA gateway). Configuration by level is simple and convenient. Manual configuration of agents may be performed to fine tune agent behavior and packet analysis.

Note that numerous SCADA protocols are used in industrial environments. Moreover, in addition to standard protocols, e.g., Modbus and DNP3, some deployments implement variations or subsets of standard protocols or proprietary protocols. The requirement to deal with diverse SCADA protocols has motivated the design of modular agents with configurable synopsis engines.

6. SCADA Traffic Storage and Querying

Forensic agents submit their SCADA traffic synopses to a designated data repository for storage. The design uses a relational database and query mechanisms to support forensic investigations. The SCADA traffic storage and querying facility incorporates a connection machine, data

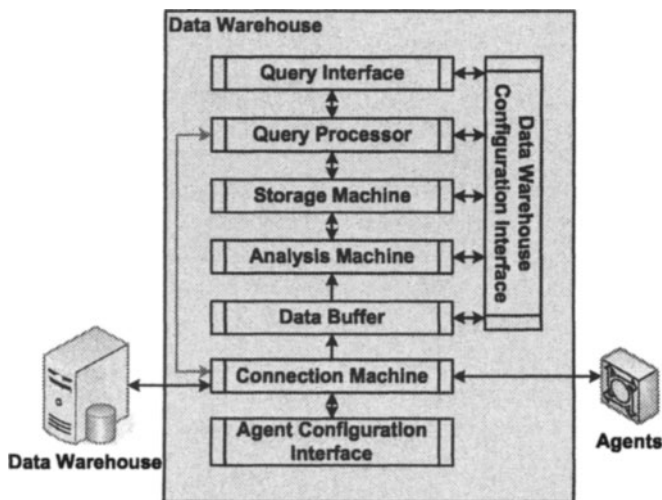


Figure 5. SCADA traffic storage and querying.

buffer, analysis machine, storage machine, query interface, query processor and agent configuration interface (Figure 5).

The connection machine supports communications between data warehouses and registered agents. Connections with registered agents are used to receive synopses and configure agents. Connections to other data warehouses facilitate the processing of queries that span multiple SCADA networks.

Synopses submitted by agents for storage are placed in the data buffer. Packet synopses are then passed to the analysis machine using a producer/consumer algorithm.

The analysis machine produces signatures associated with synopses that are used for event reconstruction as well as to analyze and correlate SCADA traffic patterns. Signatures reduce storage requirements while maintaining forensic capabilities. To illustrate the concept, consider a PLC that communicates with specific field devices. Therefore, only the relevant addresses must be stored and associated with this PLC. The corresponding device-based signature is generated by correlating synopses from all the agents that observe traffic associated with the PLC.

Pattern analysis capabilities are currently being developed for the forensic architecture. For example, PLCs execute repetitive control loops that have well-defined communication patterns. These patterns can be analyzed to produce network-based signatures for forensic investigations and anomaly detection.

The storage machine uses hash tables and a relational database. Each registered agent has a set of hash tables, which are used to index repetitive signature data associated with an agent. For example, the partial synopses generated for communication between two devices with largely static-address-oriented data need not be stored more than once. Instead a pointer to the proper entry is used as the signature, which is stored in the database, to identify the communication. The number of tables associated with an agent depends on the type and quantity of synopses generated by the agent.

The query interface enables forensic investigators to reconstruct incidents, perform system checks and analyze process trends. The current interface provides two SQL-based querying mechanisms. The first uses a GUI and a pre-defined set of elements and options to support routine analyses. The second provides a console that gives analysts complete freedom to specify queries. Results are provided in the form of reports augmented with graphical information about the SCADA network, devices and agents.

The query processor fields queries received from a local query interface or from another SCADA network via the connection machine. The query processor determines if the resolution of the query involves information from a different SCADA network. In this case, a query is sent to the appropriate data warehouse, which in turn dynamically generates and processes a query that is returned to the sender.

7. Conclusions

The use of TCP/IP as a carrier protocol and the interconnection of IT and SCADA networks expose industrial processes to attack over the Internet. Novel architectures are required to facilitate forensic investigations of SCADA network incidents. An architecture that supports SCADA network forensics can also enhance industrial operations. Network forensics requires mechanisms that systematically capture relevant traffic and state information throughout the network. In the context of SCADA networks, the capture and subsequent analysis of sensor data and control actions assists in monitoring process behavior, examining trends and ultimately optimizing plant performance.

The architecture proposed in the paper is specifically designed for SCADA networks. The architecture is scalable and flexible. Also, it can handle multiple interconnected SCADA networks and diverse protocols.

References

- [1] American Gas Association, *Cryptographic Protection of SCADA Communications; Part 1: Background, Policies and Test Plan*, AGA Report No. 12 (Part 1), Draft 5 (www.gtiservices.org/security/AGA12Draft5r3.pdf), April 14, 2005.
- [2] American Gas Association, *Cryptographic Protection of SCADA Communications; Part 2: Retrofit Link Encryption for Asynchronous Serial Communications*, AGA Report No. 12 (Part 2), Draft (www.gtiservices.org/security/aga-12p2-draft-0512.pdf), May 12, 2005.
- [3] American National Standards Institute/Instrumentation Systems and Automation Society, *Security Technologies for Manufacturing and Control Systems (ANSI/ISA-TR99.00.01-2004)*, October 2004.
- [4] American National Standards Institute/Instrumentation Systems and Automation Society, *Integrating Electronic Security into the Manufacturing and Control Systems Environment (ANSI/ISA-TR99.00.02-2004)*, October 2004.
- [5] American Petroleum Institute, *API 1164, SCADA Security*, American Petroleum Institute, September 1, 2004.
- [6] M. Berg and J. Stamp, A Reference Model for Control and Automation Systems in Electric Power, Technical Report SAND2005-1000C, Sandia National Laboratories, Albuquerque, New Mexico, 2005.
- [7] S. Boyer, *SCADA: Supervisory Control and Data Acquisition (Third Edition)*, Instrumentation, Systems and Automation Society, Research Triangle Park, North Carolina, 2004.
- [8] British Columbia Institute of Technology, *Good Practice Guide on Firewall Deployment for SCADA and Process Control Networks*, National Infrastructure Security Coordination Centre, London, United Kingdom, 2005.
- [9] E. Byres, J. Carter, A. Elramly and D. Hoffman, Worlds in collision: Ethernet on the plant floor, *Proceedings of the ISA Emerging Technologies Conference*, 2002.
- [10] E. Byres, M. Franz and D. Miller, The use of attack trees in assessing vulnerabilities in SCADA systems, *Proceedings of the International Infrastructure Survivability Workshop*, 2004.
- [11] E. Byres and T. Nguyen, Using OPC to integrate control systems from competing vendors, *Proceedings of the Canadian Pulp and Paper Association Technical Conference*, 2000.

- [12] B. Fenner, G. Harris and M. Richardson, The `libpcap` Project (sourceforge.net/projects/libpcap).
- [13] J. Graham and S. Patel, Security Considerations in SCADA Communication Protocols, Technical Report TR-ISRL-04-01, Intelligent System Research Laboratory, Department of Computer Engineering and Computer Science, University of Louisville, Louisville, Kentucky, 2004.
- [14] D. Kilman and J. Stamp, Framework for SCADA Security Policy, Technical Report SAND2005-1002C, Sandia National Laboratories, Albuquerque, New Mexico, 2005.
- [15] K. Mandia, C. Prosis and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.
- [16] Modbus IDA, *MODBUS Application Protocol Specification v1.1a* (www.modbus.org/specs.php), June 4, 2004.
- [17] Modbus IDA, *MODBUS Messaging on TCP/IP Implementation Guide v1.0a* (www.modbus.org/specs.php), June 4, 2004.
- [18] National Institute of Standards and Technology, *System Protection Profile – Industrial Control Systems v1.0*, Gaithersburg, Maryland, 2004.
- [19] K. Shanmugasundaram, H. Bronnimann and N. Memon, Integrating digital forensics in network architectures, in *Advances in Digital Forensics*, M. Pollitt and S. Sheno (Eds.), Springer, New York, pp. 127-140, 2005.
- [20] K. Shanmugasundaram, N. Memon, A. Savant and H. Bronnimann, Fornet: A distributed forensics system, *Proceedings of the Second International Workshop on Mathematical Methods, Models and Architectures for Computer Network Security*, 2003.
- [21] M. Smith and M. Copps, *DNP3 V3.00 Data Object Library Version 0.02*, DNP Users Group, September 1993.
- [22] M. Smith, and J. McFadyen, *DNP V3.00 Data Link Layer Protocol Description*, DNP Users Group, June 2000.
- [23] J. Stamp, J. Dillinger, W. Young and J. Depoy, Common Vulnerabilities in Critical Infrastructure Control Systems, Technical Report SAND2003-1772C, Sandia National Laboratories, Albuquerque, New Mexico, 2003.
- [24] The White House, *Presidential Decision Directive 63: Critical Infrastructure Protection*, National Security Council, Executive Office of the President, Washington, DC, 1998.

Chapter 23

IDENTIFYING DIGITAL CAMERAS USING CFA INTERPOLATION

Sevinc Bayram, Husrev Sencar and Nasir Memon

Abstract In an earlier work [4], we proposed a technique for identifying digital camera models based on trace evidence left by their proprietary interpolation algorithms. This work improves on our previous approach by incorporating methods to better detect interpolation artifacts in smooth image parts. To identify the source camera model of a digital image, new features that can detect traces of low-order interpolation are introduced and used in conjunction with a support vector machine based multi-class classifier. Experimental results are presented for source camera identification from among multiple digital camera models.

Keywords: Image forensics, digital cameras, color filter arrays, interpolation

1. Introduction

Advances in digital technologies have given birth to very sophisticated and low-cost hardware and software tools that simplify the creation, distribution and modification of digital images. This trend has raised new challenges concerning the integrity and authenticity of digital images. It is no longer possible to take the authenticity of digital images for granted. Image forensics, in this context, is concerned with determining the source and potential authenticity of a digital image.

Although digital watermarking technologies [2] have been introduced to address this problem, their realization requires that a watermark be embedded during the creation of a digital image. Essentially, this necessitates digital cameras to have built-in watermarking capabilities. However, this approach has not been adopted by camera manufacturers. Consequently, alternative approaches must be investigated to determine the origin, veracity and nature of digital images. The problem is further complicated by the requirement that a viable solution should require

Please use the following format when citing this chapter:

Bayram, S., Sencar, H., Memon, N., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer). pp. 289–299.

minimal prior knowledge about the specific digital camera model that was used and the conditions under which the image was captured (blind image authentication). To our knowledge, there are few, if any, techniques that can achieve these goals.

The primary assumption underlying blind image authentication techniques is that all images produced by a digital camera exhibit certain characteristics that – regardless of the captured scene – are unique to the camera due to its proprietary image formation pipeline. Practically all digital cameras encode the camera model, type, date, time and compression information in the EXIF image header. But this information can be easily modified or removed, and, therefore, cannot be used for authentication.

This paper focuses on the source camera model identification problem by identifying traces of the proprietary interpolation algorithm deployed by digital cameras. In particular, we improve on our earlier work [4] by incorporating new methodologies to capture color filter array (CFA) interpolation artifacts due to low-order interpolation.

The next section discusses current approaches for addressing the image source identification problem. The following section, Section 3, briefly describes the image formation process in digital cameras. Section 4 reviews our earlier work [4] and provides details of the improved approach. Section 5 presents the experimental results, and Section 6 contains our concluding remarks.

2. Current Solutions

In our prior work [7], we studied the source camera model identification problem by identifying and selectively combining a set of image features based on image quality metrics [3] and higher-order statistics [9] of images. This approach requires the design of a classifier that captures the variations in designated image features from different digital cameras.

Another promising approach was proposed by Lukas, *et al.* [8]. In their work, an imaging sensor's pattern noise is characterized via wavelet-based image denoising. The reference noise pattern for a particular digital camera is obtained by averaging the obtained noise residual over a number of high quality JPEG images captured by the camera. Then, a given image is matched to the camera by correlating the noise pattern of the camera (which is claimed to have captured the image in question) with the individual noise pattern extracted from the image itself.

In more recent work [4], we exploited the fact that most state-of-the-art digital cameras, due to cost considerations, employ a single mosaic

structured color filter array (CFA) rather than different filters for each color component. As a consequence, each pixel in the image has only one color component associated with it, and each digital camera employs a proprietary interpolation algorithm to obtain the missing color values for each pixel.

Our approach [4] was inspired by the technique proposed by Popescu, *et al.* [11], which was developed to detect image tampering. Their rationale was that the process of image tampering often requires an up-sampling operation which, in turn, introduces periodic correlations between the image pixels. Popescu, *et al.* designated statistical measures to detect these phenomena. In [4], we applied variants of these measures to characterize interpolation algorithms deployed by digital camera models.

The technique presented in this paper improves on our approach in [4] by designating new features. Because of perceptual image quality considerations, designers must tailor the interpolation algorithm to deal with different image features (edges, texture, etc.). This requires the introduction of strong nonlinearities to the interpolation algorithm. However, for relatively smooth image parts, most well-known interpolation algorithms (e.g., bilinear and bicubic methods) ensure satisfactory quality, and more expensive algorithms are not needed. Our premise is that most proprietary algorithms deploy simpler forms of interpolation for smooth image parts. Therefore, traces of the interpolations can be captured more effectively in these portions as opposed to busy image parts where interpolation requires more careful processing. For this purpose, we utilize the results of [6], where the periodicity pattern in the second-order derivative of an interpolated signal is analyzed.

3. Image Formation in Digital Cameras

The structure and sequence of processing in the image formation pipelines of digital cameras are very similar despite the proprietary nature of the underlying technologies. Light entering the lens of a digital camera is filtered (using anti-aliasing and other filters) and focused on an array of charge-coupled device (CCD) elements, i.e., pixels. The CCD array is the primary and most expensive component of a digital camera. Each light sensing element of the CCD array integrates incident light over the whole spectrum and produces the corresponding electrical signal representation of the scene. Since each CCD element is essentially monochromatic, capturing color images requires separate CCD arrays for each color component. However, due to cost considerations, most digital cameras use only a single CCD array. Different spectral filters,

typically red, green and blue (RGB), are arranged in a pattern so that each CCD element only senses one band of wavelengths. This spectral filter pattern, or mask, is called the color filter array (CFA). The raw image collected from the array is thus a mosaic of red, green and blue pixels.

As each sub-partition of pixels only provides information about a number of green, red and blue pixel values, the missing RGB values for each pixel must be obtained through interpolation (demosaicing). The interpolation is typically carried out by applying a weighting matrix (kernel) to the pixels in the neighborhood of a missing value. Digital camera manufacturers use proprietary demosaicing algorithms that have different kernel size, kernel shape and interpolation functions. Demosaicing is followed by a processing block, which typically involves operations such as color processing and image compression to produce a faithful representation of the scene that was imaged.

Although the image formation pipeline is identical for almost all digital cameras, the exact processing details at all stages vary from one manufacturer to another, and even in different camera models from a single manufacturer. Note that many components in the image formation pipeline, e.g., lenses, optical filters, CCD arrays, are produced by a limited number of manufactures. Because of the overlap, cameras from different manufacturers may exhibit similar qualities, and this should be taken into consideration when associating image features with the digital cameras. However, the interpolation (demosaicing) algorithm and the specific CFA pattern are often unique for each digital camera manufacturer. Our technique, which is described in the next section, exploits variations in color interpolation to classify images taken by different models of digital cameras.

4. Identifying Interpolation Traces

The methodology proposed by Popescu, *et al.* [11] analyzes traces of up-sampling to identify images (or parts of images) that have undergone resizing; this is accomplished by analyzing the correlation of each pixel value to its neighbors. Since RGB channels are heavily interpolated in a typical digital camera, in [4], we proposed a similar procedure to determine the correlation structure present in each color band and classified images accordingly. Our experimental results showed that the size of the interpolation kernel and the demosaicing algorithm vary from camera to camera [7]. Furthermore, the interpolation operation is highly non-linear, making it strongly dependent on the nature of the depicted scenery. In other words, interpolation algorithms are fine-tuned to pre-

vent visual artifacts. Busy parts of an image have over-smoothed edges or poor color transitions, while smooth parts exhibit linear characteristics. Consequently, we treat smooth and non-smooth portions of images separately in our analysis.

4.1 Non-Smooth Image Parts

We employ the Expectation/Maximization (EM) algorithm to detect traces of interpolation [11]. The EM algorithm has two main steps: an expectation step followed by a maximization step. The expectation value is computed with respect to the unknown underlying variables using the current estimate of the parameters and conditioned on the observations. The maximization step then provides a new estimate of the parameters. These two steps are iterated until convergence [10].

The EM algorithm generates two outputs. One output is a two-dimensional data array called a probability map; each array entry indicates the similarity of an image pixel to one of the two groups of samples (the ones correlated to their neighbors and those that are not) in a selected kernel. Regions of the map identified by the presence of periodic patterns indicate image parts that have undergone up-sampling. The other output is the estimate of the weighting (interpolation) coefficients that designate the contribution of each pixel in the interpolation kernel.

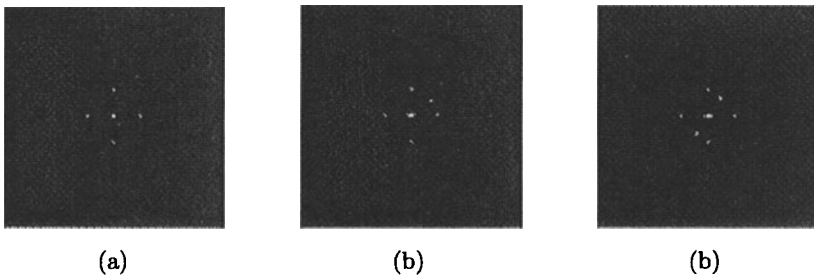


Figure 1. Frequency spectrum of probability maps obtained for (a) Nikon E-2100, (b) Sony DSC-P51, (c) Canon Powershot S200 digital cameras.

Since no *a priori* information is assumed on the size of interpolation kernel (which designates the number of neighboring components used to estimate the value of a missing color component), probability maps are obtained for varying kernel sizes. When observed in the frequency domain, these probability maps yield peaks at different frequencies with varying magnitudes, indicating the structure of the correlation between the spatial samples. Our classifier relies on two sets of features: the set of weighting coefficients obtained from an image, and the peak locations and magnitudes in the frequency spectrum. Figure 1 presents sample

magnitude responses of the frequency spectrum of probability maps for three cameras (Sony, Nikon and Canon). The three responses clearly differ in their peak locations and magnitudes.

4.2 Smooth Image Parts

Gallagher [6] showed that low-order interpolation introduces periodicity in the variance of the second-order derivative of an interpolated signal, which can be used to determine the interpolation rate and algorithm. The interpolation detection algorithm first obtains the second-order derivative of each row and averages it over all rows. When observed in the frequency domain, the locations of the peaks reveal the interpolation rate and the peak magnitudes determine the interpolation method.

We employ a similar methodology to characterize the interpolation rate and the interpolation algorithm employed by a digital camera. Most digital cameras encode and compress images in JPEG format. Due to 8×8 block coding, the DC coefficients may also introduce peaks in the second-order derivative implying the presence of some form of interpolation operation at a rate of 8. Therefore, the peaks due to JPEG compression have to be ignored when attempting to identify interpolation algorithm.

Figure 2 displays the magnitudes of the frequency response for the three models of digital cameras considered in this study. The variations in magnitude indicate that differences exist in the deployed interpolation algorithm. Therefore, the features extracted from each camera include the peak locations (except those due to JPEG compression), their magnitudes, and the energy of each frequency component with respect to other frequency components at all color bands.

5. Experimental Results

An SVM classifier was used to test the effectiveness of our technique. Several SVM implementations are available; we used the LibSVM package [5]. We also used the sequential forward floating search (SFSS) algorithm to select the best features from a given set of features.

In the first set of experiments, we used the Sony DSC-P51 and Nikon E-2100 camera models. The two cameras both have resolutions of 2 mega-pixels. The pictures are of size 1600×1200 pixels and are obtained with maximum resolution, auto-focus, and other settings at default values. To reduce the dependency on the scenery being viewed, we used pictures of the same scene that were taken by two cameras.

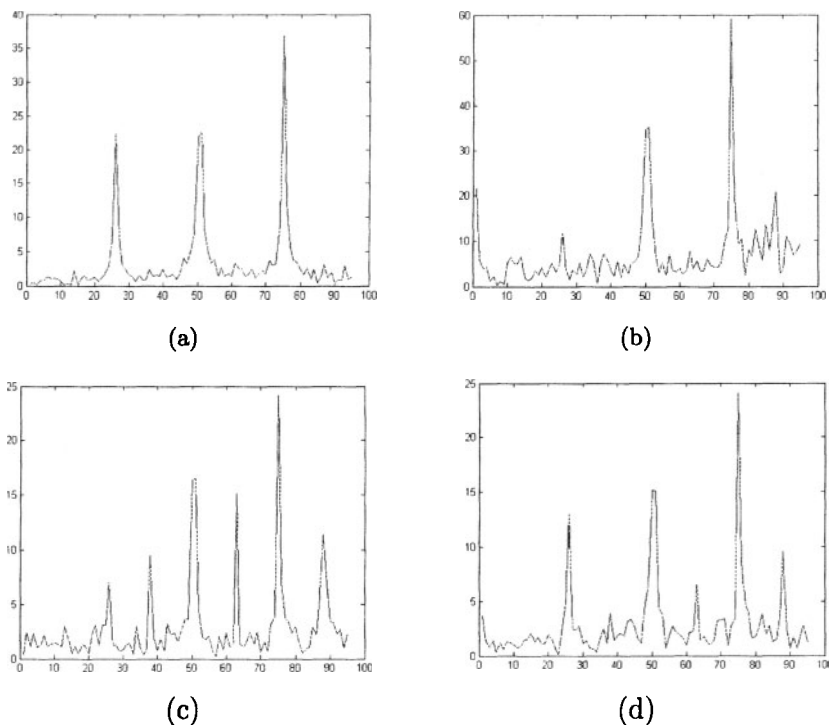


Figure 2. Frequency spectrum of averaged second-order derivatives corresponding to (a) JPEG compression and the three models of digital cameras, (b) Canon Powershot S200, (c) Sony DSC-51, (d) Nikon E-2100 with JPEG output images.

A picture data set was created by capturing 140 images with each camera model. One third of these images were used to train the classifier. The trained classifier was then used to classify the remaining two-thirds of the images. We used 75×75 pixel parts of the images in the experiments. An exhaustive search algorithm was used to partition images into smooth and non-smooth parts based on the variance of each block.

First, we extracted features assuming a 3×3 interpolation kernel for the Sony and Nikon digital cameras. The accuracy was measured as 89.3%. Next, we extracted the features considering the neighboring 4×4 pixels; the correspondingly detection accuracy increased to 92.86%. Finally, the same experiment was repeated for 5×5 neighborhoods, which produced an accuracy of 95.71%.

The three corresponding confusion matrices are presented in Tables 1, 2 and 3, respectively. The data in the tables show that accuracy improves for larger kernel sizes. These results suggest that the actual

size of the interpolation kernel used for CFA interpolation is not smaller than the considered sizes, which was empirically known to be true [7].

Table 1. Confusion matrix for two cameras (3×3 interpolation kernel).

		Predicted	
		Nikon	Sony
Actual	Nikon	95.7%	4.3%
	Sony	17.1%	82.9%

Table 2. Confusion matrix for two cameras (4×4 interpolation kernel).

		Predicted	
		Nikon	Sony
Actual	Nikon	91.4%	8.6%
	Sony	5.7%	94.3%

Table 3. Confusion matrix for two cameras (5×5 interpolation kernel).

		Predicted	
		Nikon	Sony
Actual	Nikon	94.6%	5.4%
	Sony	3.6%	96.4%

Table 4. Confusion matrix for two cameras (periodicity in second-order derivatives).

		Predicted	
		Nikon	Sony
Actual	Nikon	86.9%	13.1%
	Sony	23.3%	76.7%

Similar results were obtained for the smooth image parts using the features based on periodicity in the second-order derivatives. Table 4 shows the accuracy for the two camera case. Note that the latter set of features is not as reliable as the former set of features.

To examine how the proposed features perform for the case of three cameras, we added a Canon Powershot S200 camera to the set of cameras being investigated. The picture set for the Nikon, Sony and Canon

Table 5. Confusion matrix for three cameras (5×5 interpolation kernel).

		Predicted		
		Nikon	Sony	Canon
Actual	Nikon	85.7%	10.7%	3.6%
	Sony	10.7%	75.0%	14.3%
	Canon	0.0%	10.7%	89.3%

Table 6. Confusion matrix for three cameras (periodicity in second-order derivatives).

		Predicted		
		Nikon	Sony	Canon
Actual	Nikon	76.8%	8.9%	14.3%
	Sony	12.5%	76.8%	10.7%
	Canon	19.6%	10.7%	69.6%

cameras included various scenery images downloaded from the Internet. We extracted the features described in Sections 3.1 and 3.2 and used SVM and SFSS to classify the three cameras. An accuracy of 83.33% was obtained when features were extracted from 5×5 neighborhoods; the corresponding confusion matrix is provided in Table 5. As shown in Table 6, the accuracy dropped to 74.3% when we attempted to discriminate cameras on the basis of features obtained from smooth image parts.

Table 7. Confusion matrix for three cameras (combined set of features).

		Predicted		
		Nikon	Sony	Canon
Actual	Nikon	94.8%	1.5%	3.7%
	Sony	2.1%	95.3%	2.6%
	Canon	0.0%	2.3%	97.7%

Finally, we combined the two sets of features and repeated the experiment. In this case, the discrimination accuracy increased to 96% for the three camera case as shown in Table 7. The increase in accuracy indicates that the two sets of features capture different characteristics of an image, enabling better identification of the source camera model.

6. Conclusions

The technique proposed in this paper improves on our previous approach to source camera model identification. To detect traces of color interpolation (artifacts) in RGB color channels, we incorporate several features tuned to capture the periodicity in second-order derivatives from the features obtained using the EM algorithm [4]. A classifier is then designed using the combined set of features and tested to determine the reliability of the selected features in discriminating the source camera model from among two and three cameras. The results are promising; however, the technique is limited to images that are not heavily compressed because compression artifacts suppress and remove spatial correlations between pixels due to CFA interpolation.

References

- [1] J. Adams, K. Parulski and K. Spaulding, Color processing in digital cameras, *IEEE Micro*, vol. 18(6), pp. 20-29, 1998.
- [2] A. Akansu, E. Delp, T. Kalker, B. Liu, N. Memon, P. Moulin and A. Tewfik, Special Issue on Data Hiding in Digital Media and Secure Content Delivery, *IEEE Transactions on Signal Processing*, vol. 41(6), 2003.
- [3] I. Avcibas, N. Memon and B. Sankur, Steganalysis using image quality metrics, *IEEE Transactions on Image Processing*, vol. 12(2), pp. 221-229, 2003.
- [4] S. Bayram, H. Sencar, N. Memon and I. Avcibas, Source camera identification based on CFA interpolation, *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, pp. 69-72, 2005.
- [5] C. Chang and C. Lin, LibSVM: A library for support vector machines, version 2.81 (www.csie.ntu.edu.tw/~cjlin/libsvm), November 20, 2005.
- [6] A. Gallagher, Detection of linear and cubic interpolation in JPEG compressed images, *Proceedings of the Second Canadian Conference on Computer and Robot Vision*, pp. 65-72, 2005.
- [7] M. Kharrazi, H. Sencar and N. Memon, Blind source camera identification, *Proceedings of the IEEE International Conference on Image Processing*, pp. 709-712, 2004.
- [8] J. Lukas, J. Fridrich and M. Goljan, Determining digital image origin using sensor imperfections, *Proceedings of the SPIE*, vol. 5685, pp. 249-260, 2005.

- [9] S. Lyu and H. Farid, Detecting hidden messages using higher-order statistics and support vector machines, *Proceedings of the Information Hiding Workshop*, 2002.
- [10] T. Moon, The expectation-maximization algorithm, *IEEE Signal Processing Magazine*, vol. 13, pp. 47-60, November 1996.
- [11] A. Popescu and H. Farid, Exposing digital forgeries by detecting traces of re-sampling, *IEEE Transactions on Signal Processing*, vol. 53(2), pp. 758-767, 2005.

Chapter 24

FORENSIC ANALYSIS OF BIOS CHIPS

Pavel Gershteyn, Mark Davis and Sujeet Shenoi

Abstract Data can be hidden in BIOS chips without hindering computer performance. This feature has been exploited by virus writers and computer game enthusiasts. Unused BIOS storage can also be used by criminals, terrorists and intelligence agents to conceal secrets. However, BIOS chips are largely ignored in digital forensic investigations. Few techniques exist for imaging BIOS chips and no tools are available specifically for analyzing BIOS data.

This paper focuses on the Award BIOS chip, which is commonly used in IBM compatible machines. It demonstrates how data may be concealed within BIOS free space and modules in a manner that makes it accessible using operating system commands. Furthermore, forensically sound techniques are described for detecting and recovering concealed data from BIOS chips.

Keywords: BIOS chips, Award BIOS, data concealment, evidence recovery

1. Introduction

The Basic Input/Output System (BIOS) is the lowest level of software in any embedded device or computer [4, 13, 16, 20]. A BIOS typically resides on the motherboard within a read/write flash memory chip [8] of capacity 128K to 512K. It interfaces the hardware with the operating system, which is critical during the booting process. Also, it provides diagnostics and utilities for the computer system. A BIOS is motherboard-specific, allowing the operating system to load from and use specific hardware configurations. It maintains several system settings, e.g., drive boot order and boot-up password protection. The BIOS settings are stored separately in CMOS memory (not flash memory), which requires a small battery to maintain its integrity [20]. After the operating system loads, the BIOS passes control to the operating system.

Please use the following format when citing this chapter:

Gershteyn, P., Davis, M., Shenoi, S., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 301–313.

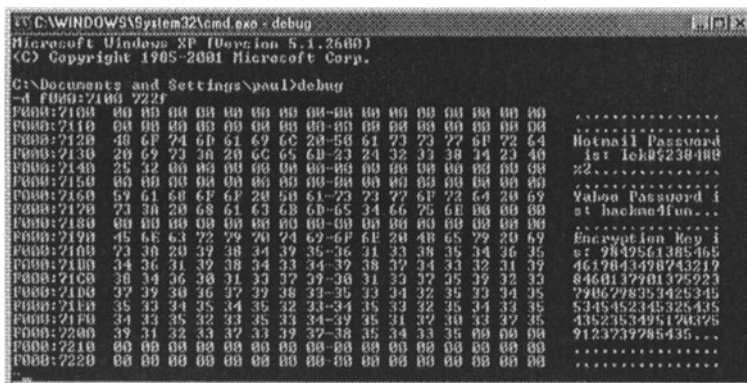


Figure 1. Hidden data in a BIOS viewed using Windows XP.

BIOS chips may contain 25K to 100K or more of unused space that can be used to store data without hindering computer performance. Unused BIOS storage space was exploited by the 1998 Win95/CIH virus that wiped out hard drives. Computer game enthusiasts often overwrite BIOS data to create personalized graphics. The same BIOS storage techniques can be used by criminals, terrorists and intelligence agents to conceal secrets, e.g., address books, financial data, incriminating photographs and cryptographic keys.

Data concealed in a BIOS chip can be accessed relatively easily. Figure 1 shows hidden data stored in the D3VA1323.BIN module of an Award BIOS chip viewed using the Windows XP command prompt. This is possible because most of the data in D3VA1323.BIN is copied to RAM during the boot process. Thus, the hidden data is discernible in a memory dump produced by the *debug* tool.

Although BIOS chips can conceal significant amounts of data, they are largely ignored in digital forensic investigations. Indeed, very few techniques exist for imaging BIOS chips [10] and no tools are available specifically for analyzing BIOS storage.

This paper focuses on the Award BIOS chip, which is commonly used in IBM compatible machines. It demonstrates how data may be concealed within BIOS free space and modules in a manner that makes it accessible using the operating system. Also, the paper suggests modifications to standard digital forensic procedures to include BIOS (and other firmware) chips.

The following section provides an overview of the Award BIOS chip, including the boot process and storage organization. Next, procedures are described for concealing data in various locations within an Award

BIOS chip. Finally, forensically sound techniques are specified for detecting and recovering hidden data from Award BIOS chips.

2. Award BIOS Overview

This paper focuses on the Award Version 6 BIOS chip (EPoX EP-D3VA with BIOS Version EPoX EP-D3VA ID# 03/23/2001-694X-596B-977-6A6LJPABC), which is representative of the family of IBM PC compatible BIOS chips. The BIOS chip contains modular software that facilitates communication between a specific motherboard and the operating system. The BIOS software runs from the BIOS chip at power-up and performs all the tasks necessary for the operating system to load successfully. The software also provides diagnostic and configuration tools for the user and low-level hardware routines for the operating system [9]. BIOS software is stored in flash memory, which allows the software to be upgraded. BIOS configuration data is stored on a separate CMOS chip. Award BIOS software consists of compressed modules along with executable code, which decompresses the modules and also provides for error recovery.

BIOS executable code usually becomes inactive after the operating system's hardware drivers are initialized. However, the BIOS may retain limited control over low-level functions such as power management.

The following subsections describe the boot process and storage organization of the Award BIOS chip.

2.1 BIOS Boot Process

A BIOS chip is critical to booting a computer. When a computer is turned on, the processor reads instructions from memory location `0xFFFF0` [13], which contains a jump call to the start of the BIOS program on the BIOS chip. When the BIOS is invoked, it executes a Power-On Self-Test (POST) that systematically checks that the necessary hardware is present and is in working order. At this point the video card is not yet initialized, so errors are communicated to the user through a series of beeps known as "beep codes" [1]. The BIOS also copies itself into system RAM for faster access, decompressing its modules in the process [7].

After the POST is completed, the system BIOS program finds and executes the video-card's own built-in BIOS program that initializes the video card. The system BIOS then locates and executes the BIOS programs of other devices.

Following these invocations, the system BIOS performs additional hardware tests and conducts a system inventory; this establishes hard-

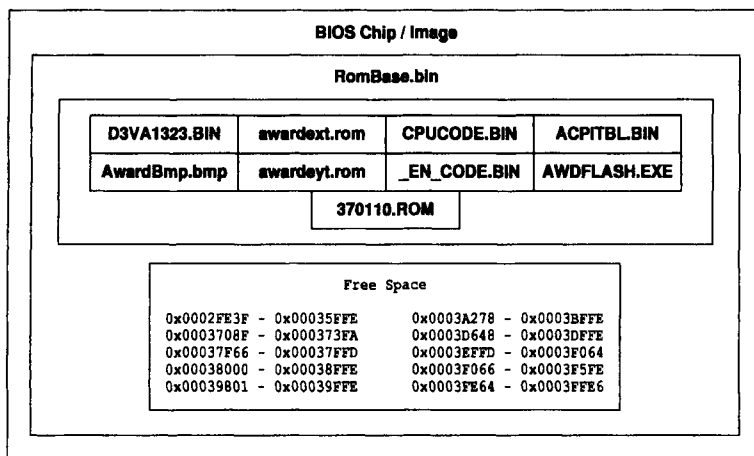


Figure 2. Award BIOS storage organization.

ware parameters. Next, the BIOS detects the memory size and identifies the drives. A summary of the system configuration is then displayed to the user.

The BIOS subsequently identifies its target boot drive, which is determined by the BIOS settings. Next, it searches for a master boot record and, upon finding it, starts loading the operating system. After the operating system is loaded, control passes from the BIOS to the operating system.

2.2 BIOS Storage Organization

The storage organization of an Award BIOS chip is shown in Figure 2. The BIOS has nine modules (D3VA1323.BIN, awardext.rom, CPUCODE.BIN, ACPITBL.BIN, AwardBmp.bmp, awardeyt.rom, _EN_CODE.BIN, AWDFLASH.EXE and 370110.ROM), which are stored at the start of the flash memory. The modules are followed by executable BIOS code and data interspersed with free space. Also, the BIOS has ten sections of consecutive free space (Figure 2). Some of these sections comprise several consecutive “blocks” of free space; each block only contains hex strings of 00s and FFs.

The integrity of all the modules as well as the first and second blocks of free space is protected by an 8-bit checksum. The integrity of the third block of free space is protected by a second checksum. However, all the free space following the third block is unprotected.

The checksum data are stored after the third block of free space at memory addresses 0x37FFE - 0x37FFF. All the data stored after address

0x37FFF is not protected in any way. A checksum mismatch is treated as a fatal error by the BIOS, which halts the booting process.

All the BIOS modules and most of the executable code and data are stored in a compressed format using the LHA/LH5 algorithm [5]. The BIOS chip incorporates code that decompresses the data during the booting process.

Data may be concealed within BIOS free space and modules. Depending on the location and amount of data concealed, the BIOS could remain functional or it could become corrupted, which prevents the computer from booting. BIOS chips are designed to be expandable. Consequently, they have large amounts of free space that can be overwritten with data without affecting BIOS operation.

BIOS modules contain text strings that are displayed as messages, e.g., error messages and hardware data. These text strings can be overwritten with data without affecting the BIOS.

Of course, the entire BIOS memory can be used to conceal data. This makes the recovery of the data problematic, as the computer cannot be booted with a corrupt BIOS. However, special devices and techniques exist for booting such a computer and recovering hidden data [10].

3. Corrupted BIOS Recovery Technique

Editing or "flashing" BIOS modules, free space or any other code/data can corrupt a BIOS. Therefore, a BIOS data hiding strategy must incorporate a technique for recovering from BIOS flashing errors.

Upon detecting an error, a BIOS chip attempts to re-flash itself using a backup BIOS file from the floppy drive. However, automatic re-flashing is unreliable because errors sometimes go undetected. Also, the data hiding process can corrupt the flashing code (AWDFLASH.EXE) itself.

A BIOS Savior device can be used to recover from a data hiding attempt that results in a corrupted BIOS. This device provides a backup BIOS chip and a hardware switch that enables the user to select whether the computer will use the BIOS Savior chip or the original BIOS chip for the booting process. The BIOS Savior plugs into the BIOS chip's socket, and the original BIOS chip plugs into the BIOS Savior. Therefore, if the BIOS on the original chip is corrupted, the user can boot the computer using the BIOS Savior, switch back to the original chip after the computer is operational, and then flash the original chip.

Before a BIOS is edited, a backup copy of the BIOS must be preserved within the BIOS Savior. This backup copy enables the computer to boot successfully regardless of the changes made to its BIOS. Note

that the booting process involves three main steps: POST, hardware initialization and master boot record access.

After data is written to the BIOS during the process of data hiding (see Section 4), two possibilities exist. The first is that the overwritten BIOS will boot the computer successfully. If the BIOS checksum computed during the POST is incorrect, the BIOS will attempt to re-flash itself. If the POST is unsuccessful and the BIOS checksum is correct, a fatal error occurs and the system halts. Regardless of the situation, the backup BIOS maintained in the BIOS Savior can be used to boot the computer. This is accomplished simply by flipping the switch on the BIOS Savior.

4. Hiding Data in BIOS Chips

This section describes procedures for hiding data in: (i) BIOS free space, (ii) BIOS modules, and (iii) free space within BIOS modules. In all three cases, substantial amounts of data can be concealed within the BIOS without hindering computer performance.

Hiding data on a BIOS chip requires a separate workstation to edit and store BIOS image files and to prepare the boot disk. Caldera Dr-DOS [2, 3] is used to create the boot disk and to facilitate BIOS flashing because it does not contain any TSR (terminate and stay resident) programs. A BIOS Savior device [12] is used to recover from BIOS flashing errors (Section 3). A hex editor is used to modify BIOS images and modules. AwardMod software [11] is used to load and store binary BIOS files and directories containing extracted BIOS modules. Also, Uniflash [17], a universal BIOS flashing utility, is used for BIOS read/writes instead of the standard Award BIOS program (AWDFLASH.EXE), which leaves portions of the chip unflashed.

4.1 Hiding Data in BIOS Free Space

The Award BIOS has 38,020 bytes of free space (located after the block of compressed modules) that can be used for storing data. In reality, 44,163 bytes of free space exist, but 6,143 bytes cannot be used because data stored in certain locations is not retained.

The Award BIOS has 12 blocks of free space; the specific locations of the blocks are shown in Table 1. Note that the first and last null bytes (00 or FF) of each free space block are assumed to belong to the code preceding/following the free space, and are therefore not counted.

Free space blocks are null blocks containing long strings of 00s or FFs (Table 1). These free space blocks can be overwritten without corrupting the BIOS. However, it is important to ensure that data written to

Table 1. Award BIOS free space blocks.

Block	Pattern	Range	Comments
Block 1	FF	0x2FE3F–0x35FFE	Protected by Checksum 1
Block 2	00	0x3708F–0x373FA	Protected by Checksum 2
Block 3	FF	0x37F66–0x37FFD	Protected by Checksum 3
—	—	0x37FFE–0x37FFF	
Block 4	FF	0x38000–0x38FFE	
Block 5	FF	0x39801–0x39FFE	
Block 6	FF	0x3A278–0x3A744	
Block 7	00	0x3A745–0x3AFFF	0x3A800–0x3AFFF not recoverable
Block 8	FF	0x3B000–0x3BFFE	Entire block not recoverable
Block 9	00	0x3D648–0x3DFFE	
Block 10	00	0x3E0FD–0x3F064	
Block 11	00	0x3F066–0x3F5FE	
Block 12	00	0x3FE64–0x3FFE6	

Blocks 1, 2 and 3 does not alter the checksums [19]. This is accomplished by reserving one byte each in Block 2 and Block 3 to balance the checksums. First, the 8-bit checksum of BIOSback.bin is computed before any changes are made (the checksum value for the Award BIOS is EA). Next, one byte in Block 2 is reserved by changing it to 00. Then, data is written to Blocks 1 and 2, and checksum is re-calculated. Finally, the reserved byte in Block 2 is changed to a value that makes the checksum equal to EA. This “balancing value” is computed as $[(Original\ Value) - (Current\ Value) + 0x100] \bmod 0x100$. Block 3 is overwritten with data and the corresponding Checksum 2 is balanced in a similar manner.

The following procedure specifies the steps involved in hiding data in BIOS free space.

BIOS Free Space Overwriting Procedure

1. Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.
2. Copy the Uniflash program (UNIFLASH.EXE) and all its required components to the boot disk.
3. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke UNIFLASH.EXE. Backup the original BIOS to the boot disk as BIOSback.bin. Copy BIOSback.bin to the workstation hard drive.
4. Use the hex editor to write data to free space in BIOSback.bin, making sure that Checksums 1 and 2 are preserved using balancing values. Save the changes in a new file called BIOSedited.bin.

5. Complete the process of hiding data by copying `BIOSedited.bin` to the boot disk. Boot the Award BIOS machine using the boot disk, and flash the BIOS chip by invoking `UNIFLASH.EXE`.
6. Restart the computer to verify that it functions properly.

4.2 Hiding Data in BIOS Modules

The following procedure lists the steps involved in hiding data in BIOS modules.

BIOS Module Overwriting Procedure

1. Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.
2. Copy the Uniflash program (`UNIFLASH.EXE`) and all its required components to the boot disk.
3. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke `UNIFLASH.EXE`. Backup the original BIOS to the boot disk as `BIOSback.bin`. Copy `BIOSback.bin` to the workstation hard drive.
4. Use AwardMod to extract modules in `BIOSback.bin` and store them in a directory named `BIOSBackup`.
5. Use the hex editor to overwrite module data that is not critical to the operation of the BIOS (e.g., text strings). This data can be overwritten with text or binary data.
6. After one or more modules are overwritten, use AwardMod to load all the files in the `BIOSBackup` directory and store them in a new BIOS image called `BIOSedited.bin`.
7. Preserve Checksum 1 in `BIOSedited.bin` using a balancing value in Block 1 or 2.
8. Complete the process of hiding data by copying `BIOSedited.bin` to the boot disk. Boot the Award BIOS machine using the boot disk, and flash the BIOS chip by invoking `UNIFLASH.EXE`.
9. Restart the computer to verify that it functions properly.

4.3 Hiding Data in BIOS Module Free Space

This section describes how data may be stored in the BIOS module `D3VA1323.BIN` in a manner that makes it accessible from Windows using the `debug` command.

Module `D3VA1323.BIN`, which contains hardware-specific settings and routines, has 5,057 bytes of free space that can be accessed from Windows using the `debug` command. The `debug` memory dump of locations

0xF0000 - 0xFFFFF contains some data from D3VA1323.BIN. Note that **debug** uses the *segment:offset* memory notation [18]; the corresponding absolute memory address notation is $16 * \text{segment} + \text{offset}$. For example, the *segment:offset* address F000:027E in RAM is equivalent to the absolute address 0xF027E in RAM. The following are the mappings of memory addresses in **debug** notation to absolute addresses in the D3VA1323.BIN module (when viewed as a file):

```
F000:027E - F000:13FF <=> 0x1027E - 0x113FF
F000:1514 - F000:1BFF <=> 0x11514 - 0x11BFF
F000:1C9F - F000:FFFF <=> 0x11C9F - 0x1FFFF
```

The last memory mapping contains two blocks of free space that constitute 5,057 bytes of 00s. Note that the last four digits of each pair of starting and ending address are identical, which simplifies the task of determining the memory addresses to be dumped to obtain the contents of a certain location in D3VA1323.BIN.

BIOS Module Free Space Overwriting Procedure

1. Run a **telnet** server on the Award BIOS machine. Use a **telnet** client to connect to the **telnet** server (this is needed to capture the screen output). Execute the command **debug**. At the **debug** prompt, type: **d F000:0000 FFFF** and press enter. Save the output in a new text document: **originalBIOS.txt**.
2. Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.
3. Copy the Uniflash program (UNIFLASH.EXE) and all its required components to the boot disk.
4. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke UNIFLASH.EXE. Backup the original BIOS to the boot disk as **BIOSback.bin**. Copy **BIOSback.bin** to the workstation hard drive.
5. Use AwardMod to extract modules in **BIOSback.bin** (see Figure 3) and store them in a directory named **BIOSBackup**.
6. Use the hex editor to view the module D3VA1323.BIN. Compare D3VA1323.BIN with **originalBIOS.txt**. Note that three hex segments of each of the two files are identical because they map to each other (see the discussion immediately preceding this procedure). Therefore, data hidden in these locations in D3VA1323.BIN can be viewed using the **debug** command.
7. The third segment of D3VA1323.BIN (i.e., 0x11C9F - 0x1FFFF), which can be viewed using the **debug** command, contains two blocks of 00s (0x16FFB - 0x17FFE, 0x1DC42 - 0x1DFFE) that can be overwritten without corrupting the BIOS. Overwrite these blocks with data that is to be hidden.
8. After the module is edited, use AwardMod to load all the files in the **BIOSBackup** directory and store them in a new BIOS image called **BIOSedited.bin**.

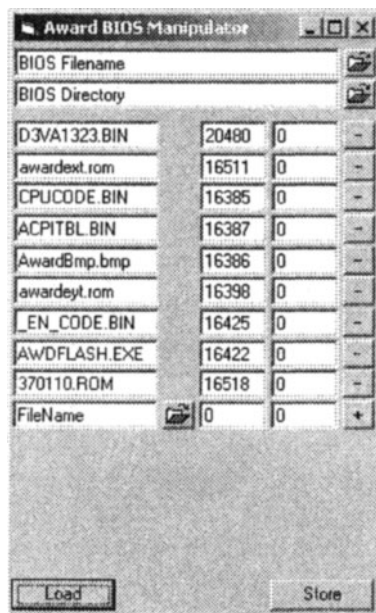


Figure 3. AwardMod screen during extraction of BIOS modules.

9. Preserve Checksum 1 in `BIOSedited.bin` using a balancing value in Block 1 or 2.
10. Complete the process of hiding data by copying `BIOSedited.bin` to the boot disk. Boot the Award BIOS machine using the boot disk, and flash the BIOS chip by invoking `UNIFLASH.EXE`.
11. Restart the computer to verify that it functions properly. Verify that the data hidden in `D3VA1323.BIN` can be viewed using the `debug` command.

5. Forensic Examination of BIOS Chips

A BIOS chip is a convenient location for hiding secrets because significant amounts of data are easily stored and retrieved. Law enforcement agents generally overlook BIOS chips during investigations. Moreover, at this time, no established forensic procedures exist for imaging and analyzing BIOS chips.

This section describes how common forensic tools can be used to examine BIOS chips. In particular, searches based on regular expressions and file headers (e.g., file carving) can be used to identify and extract data concealed in a BIOS.

BIOS Data Recovery Procedure

1. Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.
2. Copy the Uniflash program (**UNIFLASH.EXE**) and all its required components to the boot disk.
3. Boot the Award BIOS machine using the boot disk. After Caldera Dr-DOS has booted, invoke **UNIFLASH.EXE**. Backup the seized BIOS to the boot disk as **BIOSevidence.bin**; this creates a forensic image of the seized BIOS. Copy **BIOSevidence.bin** to the workstation hard drive.
4. Use AwardMod to extract modules in **BIOSevidence.bin** and store them in a directory named **BIOSevidence**.
5. Use forensic tools (e.g., Foremost, EnCase, Forensic Tool Kit, ILook) to examine **BIOSevidence.bin** and the extracted modules, especially **D3VA1323.BIN**, for text, file headers and regular expressions, and preserve all data of interest. Also, examine **D3VA1323.BIN**, which is 128K in size, manually using the hex editor to detect all hidden text.
6. If hidden data cannot be found using the forensic tools, use the hex editor to compare modules from the seized BIOS with those from a clean copy of the BIOS image (e.g., one obtained from the motherboard manufacturer). This assists in locating hidden data.
7. Use forensically sound procedures to copy and preserve all data of interest.

6. Modifications to Forensic Procedures

Traditional digital forensic investigations involve three main steps: initial response, media duplication (imaging) and imaged media analysis. Investigations are jeopardized when important evidence is stored in media that are not seized by investigators or when the media are seized but, for a variety of reasons, evidence is not recovered from the media.

The initial response step is typically executed on a live computer system that contains volatile information. This volatile information, e.g., current users, open sockets and running processes, is captured and saved for further investigation. Code and data – including concealed information – stored on a BIOS chip are not lost when a computer system or embedded device is powered down. Therefore, no action specific to the BIOS chip is necessary during the initial response step. Of course, initial responders must be aware that importance evidence may be hidden in the chip.

It is important that forensic examiners image a BIOS chip just as they image other media (e.g., hard drives and flash memory) during the media duplication step. The procedure for imaging a BIOS chip has been described in Section 5.

Some authors (e.g., [15]) recommend that digital forensic examiners view drive geometry data in the system BIOS configuration – before the media duplication step – to obtain drive parameters that might aid in media duplication. An analysis of the system BIOS configuration may reveal that data is hidden in the BIOS. However, the examiner must be alert to the fact that the BIOS may contain hidden data.

It is possible that the BIOS in a seized computer may be intentionally corrupted, e.g., when the BIOS contains secret information or when the owner has overwritten the BIOS to hinder the forensic investigation. Such a computer will not boot. Therefore, the examiner may use a chip programmer [16] to image the BIOS or the BIOS Savior device [12] to boot the computer and image the BIOS. The latter technique has been described in Section 3. Note that some BIOS chips are soldered directly to their motherboards, which renders the BIOS Savior technique useless and the chip programming technique risky at best.

During the imaged media analysis step, a forensic examiner would analyze a BIOS image using standard forensic tools as described in Section 5. Once again, the examiner should be aware of where data might be concealed and should conduct a thorough search of the BIOS image. The locations where data might be hidden in a BIOS chip have been described in Section 4.

7. Conclusions

Modern hardware components, such as dual-BIOS motherboards and replaceable BIOS chips, simplify the task of concealing secret information in BIOS chips. However, digital forensic practice has not kept up with advances in BIOS technology. As a result, few, if any, recognized techniques exist for detecting and extracting hidden data from BIOS chips. This paper has shown that even BIOS chips with checksum-based integrity protection can be used to conceal data. The other main contributions of this paper include a technique for detecting and extracting hidden data, and suggestions for modifying forensic examination procedures to accommodate BIOS chips.

“BIOS forensics” is an interesting area of digital forensic research. A library of known good hashes of BIOS chips would make it trivial to verify whether or not BIOS chips have been tampered. Note, however, that in modern computers, the extended system configuration data (ESCD) is typically stored on the BIOS chip, so the hash value computations would have to omit certain areas of the BIOS. Boot disks and CDs that automate the process of imaging BIOS chips would greatly benefit forensic

investigators. Likewise, forensic tools for heuristically analyzing BIOS images and detecting hidden data would be very valuable.

References

- [1] BIOS Central (www.bioscentral.com).
- [2] BIOSMods (www.biosmods.com).
- [3] Bootdisk.com (bootdisk.com).
- [4] P. Croucher, *The BIOS Companion*, Electrocutation Publishers, Calgary, Alberta, Canada, 1998.
- [5] M. Darmawan, Award BIOS reverse engineering (www.codebreakers-journal.com/viewarticle.php?id=38), 2004.
- [6] M. Darmawan, Award BIOS code injection (www.codebreakers-journal.com/viewarticle.php?id=58), 2005.
- [7] D. Dunn, BIOS basics (freepctech.com/articles/articles.php?ArticleId=122), 2002.
- [8] W. Gatliff, Implementing downloadable firmware with flash memory, in *The Firmware Handbook*, J. Ganssle (Ed.), Elsevier, Burlington, Massachusetts, pp. 285-297, 2004.
- [9] Gen-X-PC, BIOS info (www.gen-x-pc.com/BIOS_info.htm).
- [10] 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, New York, pp. 217-230, 2005.
- [11] J. Hill, AwardMod (sourceforge.net/projects/awardmod), 2002.
- [12] IOSS, RD1 BIOS Savior (www.iooss.com.tw), 2000.
- [13] C. Kozierok, System BIOS (www.pcguide.com), 2001.
- [14] K. Mandia, C. Proise and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.
- [15] G. Mohay, A. Anderson, B. Collie, O. de Vel and R. McKemmish, *Computer and Intrusion Forensics*, Artech House, Norwood, Massachusetts, 2003.
- [16] Phoenix Technologies, *System BIOS for IBM PCs, Compatibles and EISA Computers (2nd Edition)*, Addison-Wesley Longman, Boston, Massachusetts, 1991.
- [17] Rainbow Software, Uniflash (www.uniflash.org), 2005.
- [18] D. Sedory, Removing the mystery from segment:offset addressing (thestarman.dan123.com/asm/debug/Segments.html), 2004.

- [19] R. Sevko, Editing the BIOS (www.winsov.ru/sios002.php), 2003.
- [20] J. Tyson, How BIOS works (computer.howstuffworks.com/bios.htm).
- [21] A. Wong, *Breaking Through the BIOS Barrier: The Definitive BIOS Optimization Guide for PCs*, Prentice Hall, Indianapolis, Indiana, 2004.

Chapter 25

A TRAINING TOOL FOR INTERNET CRIMES AGAINST CHILDREN CASES

S. Aggarwal, B. Breeden, P. Henry and J. Mulholland

Abstract The Internet has greatly increased the vulnerability of children to those who would commit crimes against them. In response to Internet Crimes Against Children (ICAC) legislation, law enforcement agencies have dedicated resources to educate the public of the threat, respond to ongoing attacks, and assist victims. A significant trend in the investigation of ICAC cases is the proactive masquerading of law enforcement agents as vulnerable prey in Internet forums. This approach has shown great promise, and agents who have mastered it possess valuable knowledge and skills that could assist others. The Predator and Prey Alert (PAPA) system, a hardware and software suite of tools, originally developed for proactive shadowing, assistance and direct manipulation of a cyberstalking victim's computer, shows potential as a proactive forensic tool for ICAC investigations. This paper discusses the use of PAPA as a networked application to train law enforcement agents to investigate online cases involving the exploitation of children and teenagers.

Keywords: Proactive investigations, investigator training, Internet Crimes Against Children (ICAC) cases

1. Introduction

The Internet has greatly increased the free flow of information, overcoming many of the obstacles, e.g., culture, distance and time, that have kept people apart. Connectivity has become a two-edged sword, especially for the most vulnerable in society [9]. Increasing numbers of children are going online to learn, play and communicate with their friends [9]. In the past, child predators pursued children in public places; now they prowl online forums (chatrooms, peer-to-peer file sharing networks and gaming sites) looking for prey. The Office for Victims of Crime reports that teens, who are rebelling from parental authority or

Please use the following format when citing this chapter:

Aggarwal, S., Breeden, B., Henry, P., Mulholland, J., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 317–330.

dealing with issues of sexual identity, are especially susceptible to solicitations by Internet predators [3]. Internet Crimes Against Children (ICAC) include diverse offenses: attempted and consummated sexual assaults, illegal use of the Internet to transmit sexual material, direct solicitation of minors, and the production, possession and distribution of child pornography. ICAC crimes have several distinctive features [15]:

- Physical contact between the child and the perpetrator need not take place for victimization to occur.
- Repeated, long-term victimization may occur without the victim's knowledge. For example, sexually explicit photographs of children, once in the hands of child pornographers, often remain in circulation on the Internet indefinitely.
- Internet crimes transcend jurisdictional boundaries.
- According recent studies [3, 4], children may not realize that they have been victimized. Even if they do, they are not likely to disclose what happened.

Wolak, *et al.* [16] note that ICAC cases fall into three categories: (i) identified victims, (ii) undercover operations in which no child victims are involved, and (iii) child pornography. Only a fraction of ICAC incidents are reported to the authorities, e.g., law enforcement, Internet service providers, parents or children's hotlines [3]. As Medaris and Girouard [9] observe, "the Internet is a nearly perfect medium for offenders seeking children for sex. It provides privacy, anonymity and a virtually unlimited pool of unsupervised children and teenagers who may be susceptible to manipulation."

Nevertheless, as Mitchell, *et al.* [10] aptly point out, "this same anonymity is an advantage to law enforcement because it allows a 40-year-old investigator to go online posing as a 14-year-old girl. This permits law enforcement to be proactive in investigations in ways they previously could not, and it allows them to detect some offenders before they victimize an actual child."

The global reach and anonymity provided by the emergence of collaborative applications on the Internet have increased the scale and scope of ICAC cases [5]. Pedophiles currently use the Internet to :

- Establish instant worldwide access to potential child victims or other predators.
- Discuss their sexual desires openly.
- Share ideas about ways to lure victims.

- Provide mutual support of their adult-child sex philosophies.
- Assume disguised identities for approaching children.
- Participate in “teen chat rooms” to find out how and whom to target as potential victims.
- Identify and track down home contact information.
- Build long-term virtual relationships with potential victims [9].

Because of the vast number of Internet users, pedophiles easily find victims, and because of the cross-jurisdictional range of the Internet’s reach, offenders face little risk of interdiction. A U.S. Department of Justice (DoJ) report [15] notes that the Internet is being used by predators to contact children and/or teenagers for the purpose of engaging them in sexual acts. The Internet is also used to produce, manufacture and distribute child pornography. Young people are constantly being encouraged to engage in and exchange pornography.

According to Medaris and Girouard [9], “[p]ornography is used to break down inhibitions and validate sex between children and adults as normal, and it enables the offender to have power over the victim throughout the molestation. When the offender loses interest, pictures of the victim are often used as blackmail to ensure the child’s silence, and when these pictures are posted on the Internet, they become an enduring and irretrievable record of the victimization and a relentless, shame-inducing violation of that child’s privacy.”

Also, the Internet is being used to entice and exploit children with respect to sexual tourism, i.e., travel—with the intent to engage in sexual behavior—that is used either for commercial gain or personal gratification. This type of offender is not only willing to invest considerable time and effort to befriend and disarm a child, he is willing to cross jurisdictional boundaries. According to a DoJ report [3], “perpetrators travel hundreds of miles to different states and countries to engage in sexual acts with children they met over the Internet. Many of these cases involve local, state, federal and international law enforcement entities in multiple jurisdictions.”

Cases dealing with the exploitation of children for unlawful purposes have been steadily increasing [14], but the resources needed to investigate these cases have not kept pace because the investigation of ICAC cases places substantial burden on local law enforcement. Wolak, et al. [16] observe “[s]ince the mid-1990s ... developing technologies have forced law enforcement to confront situations not anticipated in criminal statutes, master technical advances, develop new investigative techniques, and

handle criminal cases that often span multiple jurisdictions.” According to Mitchell, *et al.* [10], ICAC are “widespread, occurring throughout the criminal justice system; they are multi-jurisdictional [and] so require extensive collaboration; they involve constantly changing technology; and they require specialized investigation methods.”

2. Proactive Investigations

In response to the growing challenges of online crime, law enforcement has deployed innovative techniques to combat ICAC perpetrators. Foremost among these are proactive investigations that involve law enforcement agents posing as minors, lurking in chatrooms, and waiting to be contacted by offenders seeking underage victims [7].

Law enforcement agents at all levels are conducting proactive investigations for several reasons, including increased public safety, relatively low cost and administrative ease, and the potential to stop or apprehend criminals before they can harm innocent persons [15]. Although the agents pose as entities other than themselves, this activity is not considered entrapment if agents wait for suspects to make clear statements demonstrating a predisposition to commit a crime [14]. No minors are involved in these types of cases, but because of the anonymity of online forums, suspects believe they are communicating with minors. As such, the cases are referred to as “proactive” because they allow law enforcement to act without waiting for an offender to commit a crime against a juvenile victim.

According to Mitchell, *et al.* [10], proactive cases represent 25% of all arrests for Internet sex crimes against minors. Despite the advantages of proactive investigations, given the large number of law enforcement agencies, many with limited sworn officers, it remains a significant temporal and monetary burden to train and deploy agents in these investigations. For law enforcement to effectively track, arrest and gather evidence of ICAC incidents, they must be well-versed in computers and the Internet, the online social behavior of young people and suspects, and relevant investigative techniques [2]. ICAC cases and the agencies that respond to them require financial resources to acquire, maintain and upgrade equipment, maintain staff with expertise in computer technology, provide training in specialized investigation methods, and promote inter-jurisdictional cooperation—all while technology and criminal techniques are becoming increasingly sophisticated. Ideally, ICAC investigations can stop crimes in progress; findings suggest that the Internet may allow law enforcement to act before a youth is victimized, gather solid evidence of offenses, and track offenders [2]. Mitchell, *et al.* [10] sug-

gest that offenders arrested in undercover investigations pose significant risks to young people: in 13% of undercover investigations, offenders were found to be concurrently committing similar crimes with juvenile victims leading to the identification of molested youth. In 41% of undercover investigations, offenders possessed child pornography, revealing additional criminal conduct [7]. The track record so far is good: proactive prosecution of ICAC cases has produced high rates of guilty pleas and low rates of dismissed and dropped cases.

3. Using PAPA as a Training Tool

Several law enforcement agencies have developed facilities for ICAC investigations and many agents have gained invaluable experience in proactive techniques. In the face of the widespread ICAC threat, it is desirable that they leverage their experience and expertise to fight this growing category of crime.

We argue that the Predator and Prey Alert (PAPA) system [1], a tool developed for cyberstalking cases, is well suited for ICAC cases. The core PAPA system is a set of integrated tools originally designed to support agencies in helping victims of cyberstalking, facilitating the investigation of such crimes, and collecting, verifying and maintaining evidence for the subsequent prosecution of cyberstalkers. Not only can PAPA be adapted to proactive and reactive investigations, but it can also be deployed as an in-house or distributed training tool that can be used by expert agents to oversee and instruct inexperienced agents during ongoing investigations.

PAPA was designed with the following goals in mind:

- Permit agencies to remotely “shadow” a victim and provide assurance and advice when needed.
- Capture and log circumstantial data related to stalking activities so that an analyst can subsequently investigate and determine the identity of the predator.
- Capture evidence of probative value so that the suspect can be successfully prosecuted.

The PAPA architecture is illustrated in Figure 1. It is assumed that the expert agent has a primary communication channel with the predator via the Internet where he or she is working a case from home (this is standard practice in ICAC cases) [3]. Each time the investigating agent engages with the predator, say in an online game or in a chatroom, a session is established. A session used for evidentiary purposes begins

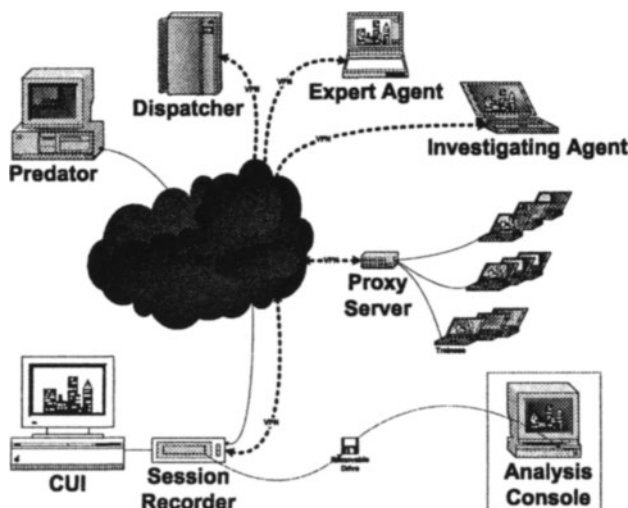


Figure 1. PAPA system architecture.

when the trainer agent logs in to guide the interactions between the predator and the investigating agent and endures until one of the parties logs out from the Computer Under Investigation (CUI). PAPA records the actual framebuffer content of the CUI during a session.

A hardware Session Recorder is connected to the CUI either through a USB or an Ethernet connection via LAN. This approach has several evidentiary advantages over collecting data directly on the CUI or sending video through slower channels. First, it minimizes modification of the CUI. Second, software-only solutions are potentially more insecure and susceptible to manipulation, and any evidence collected may become tainted. Third, it is much easier and less disruptive to transfer the large-capacity, dedicated disk from the Session Recorder to the Analysis Console while preserving the chain of custody. Fourth, private and sensitive data on the CUI that is irrelevant to the investigation is not compromised. Note that any solution that transmits captured video and other data to a remote storage location via the Internet may not be practical in low bandwidth environments.

PAPA keeps the management of captured data separate and independent of the other CUI functions. An independent second channel is used to communicate with the law enforcement expert (trainer agent). This channel could be a phone line, another high-speed Internet connection, Virtual Private Network (VPN), cellular, wireless, satellite, etc. Authorization and case coordination are achieved through communication

with the Dispatcher, which informs the expert when the investigating agent requests assistance, authenticates the agents' connections to the CUI via the Session Recorder, and continually monitors the state of the Session Recorder via a "heartbeat" protocol.

For completeness, PAPA captures all the video information from the framebuffer on the CUI. The high-speed connection permits the capture of video in raw mode, yielding a high-resolution image of activity on the CUI during the session. The Session Recorder also captures other meta information related to the session in the form of keystrokes and tagging of events. Video and metadata are time stamped, and meta-data captured by filtering communications to the investigating agent's computer, such as IP header information and predetermined auxiliary textual and temporal information in the packet data, e.g., target words, screen names, email addresses and avatars, are also stored and used to index the video files. This indexing allows for rapid analysis of the potentially large evidence files.

The Session Recorder has a second logical channel between agents and the Dispatcher. This second channel is also secured using a VPN tunnel to conceal agent traffic from the predator, who communicates on the primary Internet connection channel and may have the ability to detect unusual traffic over the primary channel. If necessary, the second channel can be implemented over the primary connection. However, the investigating agent may lose bandwidth in the first channel and the predator may be able to detect traffic in the second channel.

The Session Recorder is connected to the Dispatcher to keep track of its status, and the Dispatcher also mediates all connections between the expert and the investigating agent. The channel between the agents operates transparently through the Session Recorder and all interactions can be viewed over the independent second channel. The channel between the agents is primarily implemented through a customized version of Virtual Network Computing (VNC) open source software [13]. VNC supports a variety of remote viewing and control modes between the remote desktop of the CUI and the agent. It requires a client running on the agent's computer (Agent Module), and a modified VNC server (Victim Module) running on the CUI. The VNC-based connection permits agents to view the investigating agent's screen and to take control of the CUI when necessary. PAPA implements an additional "chat channel" between the expert and the investigating agent to allow the expert to interact with the investigating agent independent of the communication between the investigating agent and the predator. This chat channel is also implemented through the independent second channel and the Session Recorder.

The PAPA system provides several features:

- Recording the user experience of the “victim,” including all communications between agents, between the trainee and predator, and between the Session Recorder and the Dispatcher, in high-resolution, lossless and verifiable formats.
- Ensuring the integrity of evidence recorded by the Session Recorder via robust cryptographic hashing and access control mechanisms.
- Extending the evidentiary “chain of custody” from the CUI.
- Capturing all available evidence of the investigating agent’s interactions during an online session, including relevant metadata such as time stamps and potentially relevant TCP/IP packet header and data information.
- Preventing all undetected pre- and post-computing of evidentiary data.
- Indexing suspected attacks and other incidents of interest within the potentially large video files created during a session.
- Coordinating network communications between the investigating agent’s computer and law enforcement systems by means of an independent and secure second communications channel.
- Providing strong time-based verification of data reception, recording, and encryption through massive redundancy.
- Enabling flexible playback and queries of the online experience by both the agents to augment the learning experience.

This approach avoids the common issues of anonymity, lack of records and under-reporting inherent in computer crime cases. Furthermore, the predator’s expectation of anonymity helps undercover agents monitor the predator’s activities because the agents can assume the online identity of the victim to solicit further personal information and/or to set up a rendezvous with the predator.

4. PAPA Functionality

Hardware and software for recording video, audio and keystroke activity on the trainee’s computer are currently feasible. This paper focuses on the ability to legally and contemporaneously record the activities involved in ICAC cases with the goal of gathering admissible electronic

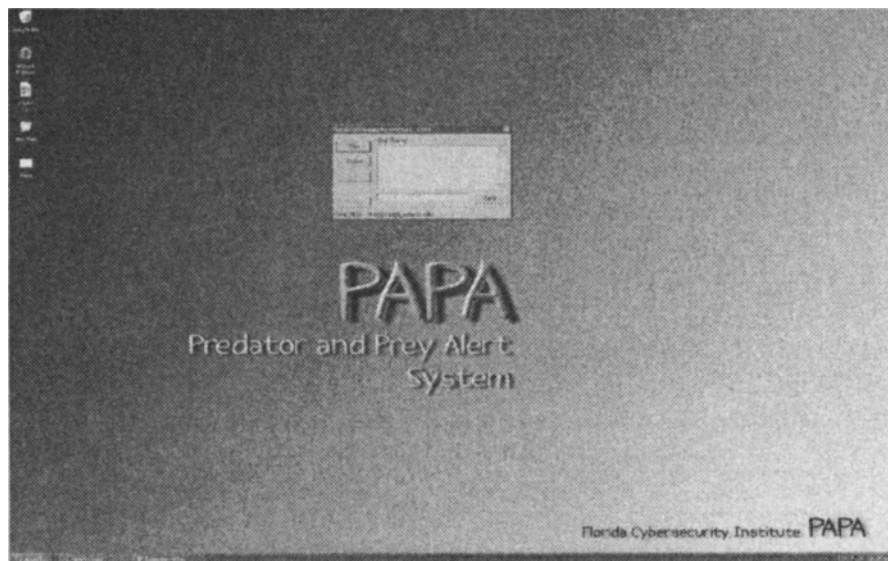


Figure 2. Expert Agent Module initialization.

evidence. All the PAPA features listed above have been designed with flexibility of deployment in mind. For example, the Session Recorder can be connected to the CUI in several ways, and the Dispatcher allows multiple agents in different locations to authenticate, log in, communicate and shadow the session.

Figure 2 shows the desktop of the Expert Agent Module, which features clients and their icons to initiate communication with the investigating agent. The initial chat dialog box is semi-transparent to allow monitoring of background windows. It always on top of the desktop so that incoming chat communication is visible. The box features buttons to view the CUI, control the CUI, tag incidents, stop the session and send messages to the CUI.

Figure 3 shows the desktop of the remote CUI and its contents in a window on the expert's computer. In this example, the CUI is running the popular game *World of Warcraft*, which demonstrates the graphic capabilities of PAPA. The game contains a fully functional chat room that is often a forum for predatory attacks. All the traffic between players is encrypted, which demonstrates PAPA's ability to do "insider" proactive investigations. In particular, note the lower video resolution within the window. The resolution is configurable from raw (full native resolution) to high levels of compression (Figure 3). This high level of compression

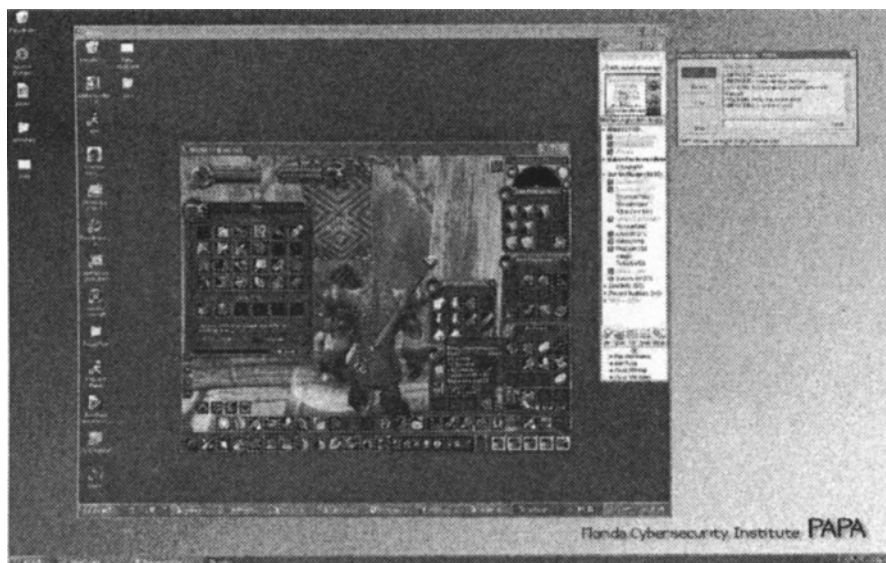


Figure 3. Expert Agent Module during a session.

allows for remote shadowing over low bandwidth connections, but comes at the cost of lower readability, stuttering screen refresh artifacts, and additional CPU load during signal compression and decoding.

Figure 4 shows the Expert Agent Module operating in View Mode, in which the agent can shadow (and record) the PAPA session. The expert may chat with the Investigating Agent Module via standard text entry, read messages from the CUI, and scroll through the dialog history. There is also the option of running PAPA in Control Mode, when the expert can preempt the local input of the CUI. Figure 4 shows a character in World of Warcraft stalking the investigating agent on the CUI. The attacker is “King Arabi” and he has typed: *“Why don’t you call me? When can I see you again?”* in the game’s chat box (lower left quadrant). The investigating agent notes that the attacker, who is not in “her” Friends List, has reappeared. The expert indicates that she sees the attacker too. At this point, the instructor guides the inexperienced agent through the session. Together, they engage the predator so that the investigating agent can solicit evidence for prosecution. In fact, agents being trained to work ICAC cases can view the entire sequence of events and discuss the effectiveness of various investigative techniques.

Figure 5 presents a view of the CUI desktop. It shows the Investigating Agent Module chatbox, which is moveable and semi-transparent.

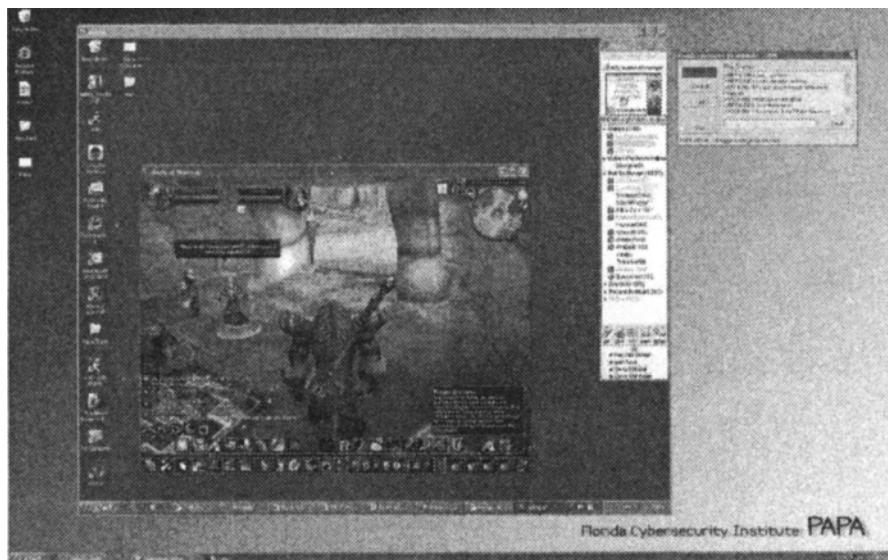


Figure 4. Evidence of a predatory attack.



Figure 5. CUI desktop during an attack.



Figure 6. Expert Agent Module view of attack.

The chatbox buttons are Record, Tag, Stop and Send. It includes a scrollable dialog box that indicates that the session is being recorded. The *World of Warcraft* game is running in a window, and the chat between players is in the lower left corner.

Figure 6 shows the expert's view of the previous scene. Incoming and outgoing traffic are indicated in a Sygate Personal Firewall window, and memory and CPU load in the Windows Task Manager window.

PAPA can send the entire sequence to multiple law enforcement agencies to support remote training. For large-scale training applications it is desirable to operate PAPA in Proxy Mode, in which a proxy server is configured to relay the video. In this mode, PAPA can support higher resolution, switch between different host connections on the fly, enable reverse host connections, and accommodate hosts with variable desktop geometries.

VNC Reflector [13], a specialized VNC server, provides the ability to work with a large number of clients (viewers). However, if the signal is compressed to accommodate limited bandwidth, the viewing agents' computers must be fast enough to handle the overhead of decompression on the fly. Our experiments indicate that a 2+ GHz Pentium 4 machine with 1 GB RAM is sufficient for this purpose.

5. Conclusions

PAPA provides law enforcement agents with an understanding of proactive techniques used in ICAC investigations. It exposes trainees to advanced investigative techniques and protocols. Even more promising is PAPA's ability to leverage the behavior of an actual "gamer" as the CUI operator. The chat functionality provides support for the discourse between trainers and trainees, and is free of spatial constraints.

PAPA can be used for a variety of proactive investigations, including cases in which agents impersonate child pornography traders [3]. Also, it can support "reactive" or "take-over" investigations—when an officer goes online posing as a youth who has been solicited or as another youth. Furthermore, PAPA can support undercover operations that typically require covert, stand-alone phone lines or Internet access through non-government service providers [8].

PAPA is an effective training tool. Using the PAPA Session Recorder, experts can record, index and archive sessions for playback, self-paced learning and translation to other languages. PAPA helps agents and trainers overcome technical impediments in online child exploitation investigations. Also, it facilitates the rapid dissemination of new proactive investigation techniques.

Acknowledgements

This work was supported by the National Institute of Justice under Grant 2004-RD-CX-K154.

References

- [1] S. Aggarwal, M. Burmester, P. Henry, L. Kermes and J. Mulholland, Anti-cyberstalking: The Predator and Prey Alert (PAPA) system, *Proceedings of the First International Workshop on Systematic Approaches to Digital Forensics Engineering*, pp. 195-205, 2005.
- [2] S. Aggarwal, P. Henry, L. Kermes and J. Mulholland, Evidence handling in proactive cyberstalking investigations: The PAPA approach, *Proceedings of the First International Workshop on Systematic Approaches to Digital Forensics Engineering*, pp. 165-176, 2005.
- [3] B. Breeden and J. Mulholland, Investigating Internet Crimes Against Children (ICAC) cases in the State of Florida, to appear in *Proceedings of the Twenty-First Annual ACM Symposium on Applied Computing*, 2006.

- [4] D. Faulkner and D. Mahoney, Brief overview of pedophiles on the web (www.prevent-abuse-now.com/pedoweb.htm), 1997.
- [5] D. Finkelhor, K. Mitchell and J. Wolak, *Online Victimization: A Report on the Nation's Youth – 2000*, National Center for Missing and Exploited Children, Washington, DC, 2000.
- [6] Fox Valley Technical College, Protecting children online (www.missingkids.com/en_US/documents/pco_agenda.pdf), 2005.
- [7] M. Girado, T. Deck and M. Morrison, Dissociative-type identity disturbances in undercover agents: Socio-cognitive factors behind false-identity appearances and reenactments, *Social Behavior and Personality*, vol. 30(7), pp. 631-644, 2002.
- [8] ICAC Task Force, Training and Technical Assistance Program (www.icactraining.org/training.htm), 2005.
- [9] M. Medaris and C. Girouard, Protecting Children in Cyberspace: The ICAC Task Force Program, NCJ 191213, Office of Justice Programs, Washington, DC (www.ncjrs.gov/html/ojjdp/jjbul2001_12_5/contents.html), 2002.
- [10] K. Mitchell, J. Wolak and D. Finkelhor, Police posing as juveniles online to catch sex offenders: Is it working? *Sexual Abuse: A Journal of Research and Treatment*, vol. 17(3), 2005.
- [11] National Center for Missing and Exploited Children, Alexandria, Virginia (www.ncmec.org), 2005.
- [12] SourceForge.net, VNC reflector (sourceforge.net/projects/vnc-reflector), 2005.
- [13] TightVNC Software, TightVNC (www.tightvnc.com), 2005.
- [14] U.S. Department of Justice, Internet Crimes Against Children, NCJ 184931, Office for Victims of Crime, Washington, DC (www.ojp.usdoj.gov/ovc/publications/bulletins/internet_2_2001/internet_2_01.html), December 28, 2004.
- [15] U.S. Department of Justice, Child victimization, *National Victim Assistance Academy Textbook*, Office for Victims of Crime, Washington, DC (www.ojp.usdoj.gov/ovc/assist/nvaa2002/toc.html), December 28, 2004.
- [16] J. Wolak, K. Mitchell and D. Finkelhor, Internet sex crimes against minors: The response of law enforcement, National Center for Missing and Exploited Children, Alexandria, Virginia (www.missingkids.com/en_US/publications/NC132.pdf), 2003.

Chapter 26

PROCESS FLOW DIAGRAMS FOR TRAINING AND OPERATIONS

Jacobus Venter

Abstract This paper focuses on the use of process flow diagrams for training first responders who execute search and seizure warrants at electronic crime scenes. A generic process flow framework is presented, and the design goals and layout characteristics of process flow diagrams are discussed. An evaluation of the process flow diagrams used in training courses indicates that they are beneficial to first responders performing searches and seizures, and they speed up investigations, including those conducted by experienced personnel.

Keywords: Process flow diagrams, first responders, search and seizure

1. Introduction

The rapid development and use of information and communications technology have influenced everyday life, mostly in a positive manner. However, the technology is also being exploited for criminal purposes. Computer-related crime is very significant—according to a recent study, the monetary impact of high tech crime in the United Kingdom in 2004 exceeded £2 billion [4]. This situation has resulted in an increased demand for personnel with digital forensics expertise in law enforcement and other government agencies.

Responding to the need for trained personnel, the South African Government instituted several digital forensics training programs. However, the task is complicated by the lack of personnel with adequate expertise or formal training in information and communications technology. This is true internationally: in the United States, for example, only a small number of investigative units have computer scientists or other technically trained individuals on staff [3]. Therefore, it is often the case that

Please use the following format when citing this chapter:

Venter, J., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 331–342.

existing personnel, even those without advanced technical skills, must be trained in digital forensics prior to working in the field.

This paper focuses on training first responders who are responsible for collecting items that may contain electronic evidence. In the South African context, first responders are specially-trained investigators or inspectors who normally act on search and seizure warrants. Before undergoing training, most of these individuals have limited understanding about electronic crime scenes and little expertise in computers and electronic devices. Therefore, it is important to develop effective training regimens for individuals with limited background and technical skills.

Electronic crime scene handling procedures are generally presented in the form of descriptions or lists (see, e.g., [2, 6, 7]). Much of the literature provides general principles, but first responders need detailed guidance. In many cases, when detailed steps are provided, they presume expertise in information and communications technology. For example, one of the steps in RFC 3227 [2] is: "For each system, obtain the relevant order of volatility." This assumes that first responders know the volatilities of systems and can prioritize them.

In our experience, this was definitely not the case for most trainees from law enforcement agencies. Some of the trainees, for example, spent inordinate amounts of time seizing CDs and then rushed through the process of seizing hard disks, which are much more important. The well-known U.S. Department of Justice guide for first responders [6] provides detailed, sequenced information, but the information is spread over several pages. We observed that first responders often did not refer to this information during seizure. Also, because the information was spread over several pages, they used some of it in the wrong context.

This paper discusses a process flow strategy, which we created to address deficiencies inherent in first responder training programs and to support operations. First, we developed a model of the tasks that had to be performed by first responders. This model was articulated in terms of process flow diagrams, which were subsequently tested in training courses offered to first responders with limited expertise in information and communications technology.

2. Motivation

Building the skills of digital forensics professionals requires ongoing attention. A 2001-02 national needs assessment on law enforcement tools and technologies conducted by the Institute for Security Technology Studies at Dartmouth College [3] noted that training programs that fit law enforcement needs were a specific requirement. This view was

reinforced in a 2004 study by Rogers and Seigfried [5], which indicated that education/training and certification were major needs. The same viewpoint was also expressed by a special investigation unit in the South African law enforcement community.

First responders need to understand the basic actions to be taken upon encountering an electronic crime scene. Several manuals have been developed to train and assist first responders. A good example is the U.S. Department of Justice guide for first responders [6], which provides excellent information on handling electronic crime scenes. However, the guide is not very useful to first responders with limited expertise in information and communications technology.

We have developed process flow diagrams to address this problem. The process flow diagrams provide structured paths of actions for first responders to follow and to check off as they are taken. Process flow diagrams also speed up investigations, including those conducted by experienced personnel.

Beebe and Clark [1] argue that digital investigation frameworks should not use a checklist approach—each situation is likely to be unique and different steps may have to be taken in each situation. Since process flow diagrams are checklist-oriented, they are subject to critique. We argue, however, that the target audience for the process flows requires a more rigorous approach that will deal adequately with most situations. This view is supported by RFC 3227 [2], which suggests that the number of decisions made during the collection process must be minimized. Additional support comes from Wolfe [9], who emphasizes that following a process decreases the probability of making errors and facilitates good documentation. Individuals without sufficient technical qualifications and experience should not deviate from the prescribed steps because they may not be able to explain the implications in courtroom testimony. We also argue that a process flow diagram adds sequence to actions in a manner that is easier to understand than a detailed list. On the other hand, we believe the objectives-based approach of Beebe and Clark [1] is well suited to complex situations and advanced phases of the forensic process. Indeed, we use process flow diagrams for situations that do not involve complex configurations or environments.

3. Design Goals and Layout Characteristics

This section discusses the design goals and layout characteristics for process flows. The actual process flow diagrams created for first responders are presented in the next section.

The first design goal was ease of use by individuals with limited expertise in information and communications technology. Digital forensics is a highly technical field, but many first responders do not have adequate knowledge and skills. Consequently, the focus was to develop process flow diagrams that would enable non-technical individuals to perform adequately in the majority of cases.

The second design goal was to create process flow diagrams applicable to the most likely cases. Developing process flows to deal with the variety of equipment, installations and configurations found at crime scenes is not be feasible. Therefore, the diagrams were built around common equipment and frequently occurring scenarios. The scenarios used to create the process flows were based on more than 50 cases.

The third design goal was to ensure that the process flow diagrams, at the very least, would not interfere with expert testimony, and, if possible, strengthen the testimony. Therefore, the design of the diagrams had to take into account potential evidentiary challenges arising from the actions of first responders. It is in this context that the checklist nature of the process flow diagrams could be problematic, especially if a checklist is not followed exactly and the first responder cannot explain why the deviation occurred. We argue that, given the target users of the process flow diagrams, it is more important to ensure that a first responder can testify confidently about the process than his ability to handle exceptions. Wolfe [10] notes that a good attorney can rattle or confuse any witness, reducing the value of the testimony or negating it altogether. He also emphasizes that close attention must be paid to following and documenting the forensic methodology [11]. Process flow diagrams support this by providing well documented methodologies for searches and seizures, and they assist first responders in reporting on their actions with confidence.

The final design goal was to create process flow diagrams that could be used during operations as well as training. Indeed, in our training sessions, many participants mentioned that they intended to use the process flows in operations.

In addition to the design goals, certain layout characteristics of the process flow diagrams are worth mentioning. The first characteristic is that each process flow must be reproducible in black and white, and should fit on a single A4 page. This layout facilitates duplication, and allows the process flow diagrams to be compiled in a normal A4 record book.

Recording information is vital throughout the forensic process, including during the evidence collection phase [2]. The process flows support this aspect by ensuring that important information is captured when

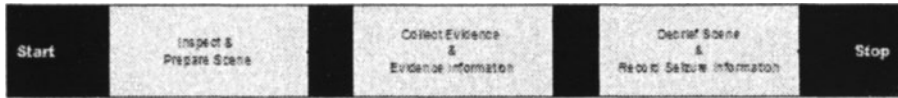


Figure 1. Generic framework elements.

following the steps in a process flow diagram. All information recorded is clearly associated with a specific case, site and room, because these are all recorded in the process flows.

Certain naming conventions are indicated in the process flow diagrams. These remind first responders about the correct naming conventions for specific pieces of evidence. For example, the process flow diagram for seizing storage media (Figure 5) employs the naming convention `Case_Site#_Room#_xxxx_EVxxx`. The descriptor used for the first `xxxx` part is “CD/DVD/STIFFY/FLASH/OTHER.” This naming convention, which is used throughout the South African system, clearly identifies the origin of evidentiary material.

4. Process Flow Diagrams

This section describes the four process flow diagrams that were developed for first responders. One process flow deals with general behavior at electronic crime scenes; the remaining three cover specific types of devices.

Figure 1 presents the generic framework elements in each of the process flows. The three elements are: “Inspect & Prepare Scene,” “Collect Evidence & Evidence Information,” and “Debrief Scene & Record Seizure Information.”

The “Inspect & Prepare Scene” element contains actions to prepare first responders for the tasks to follow (e.g., “Use Gloves” in Figures 3, 4 and 5), actions to survey the scene (e.g., “Suspect Around?” in Figure 2), actions specific to the equipment to be seized (e.g., “LAN/Modem Connected” in Figure 3), and actions to prepare the scene for the actual collection of evidence (e.g., “Write Protect Stiffy” in Figure 5). Note that in South Africa, 3.5-inch floppy disks are known as “stiffies” because they are stiffer than the older 5.25-inch floppies.

The actions in the “Collect Evidence & Evidence Information” element revolve around recording information related to specific evidentiary aspects (e.g., “Computer Information – Record and Label” in Figure 3), assigning unique identifiers to evidentiary items (e.g., “Assign evidence number, place in evidence bag” in Figure 5), and noting special infor-

mation (e.g., “Apply power and reboot machine into BIOS setup” in Figure 3).

In the “Debrief Scene & Record Seizure Information” element, actions occur to record the existence/handling over of evidentiary items (e.g., “Complete Acknowledgement of Receipt Form” in Figure 2), collect evidence in groups (e.g., “Package/Bubble Wrap Hard Disk Drives – Place into Evidence Bag” in Figure 3), and record the individuals involved (e.g., “Seizure Done by” and “Seizure Witnessed by” in Figures 3, 4 and 5). These generic framework elements create a sense of comfort for first responders because the same basic steps are followed for all categories of evidence.

The first general aspect is recording information in process flow diagrams. RFC 3227 [2] indicates that where and when the evidence was discovered and collected, and who discovered and collected the evidence must be noted. These are addressed in two ways. First, at the top of all the process flow diagrams, the CASE, SITE, ROOM, DATE and TIME details are captured. Second, information is captured within the process flows themselves. See, e.g., the space for noting the details of the individuals who performed the seizure and witnessed the seizure in Figures 2, 3 and 4.

The second general aspect is using cameras. Photographs are useful and important [6–8]. In all the process flow diagrams, photo points are shown with a  icon. Photographs assist in documenting the exact setup of the equipment, screenshot, cabling, peripheral devices, etc. They also help solve disputes that might occur later. For example, the owner of a seized computer might contend that the one presented as evidence is not the one that was actually seized. Detailed photographs taken at the crime scene would quickly resolve the issue.

Figure 2 shows the process flow diagram for responding to an electronic crime scene. This process flow begins by verifying the search warrant: it is necessary to verify that the warrant covers the applicable electronic devices searched for and seized by the first responder. The next step in the process flow is to separate all persons from computer equipment as soon as possible [9]. The rest of the process flow focuses on identifying and recording the evidence, and directs first responders to the appropriate detailed process flow diagram. The sequence, PC then PDA or cell phone, then CD/DVD, stiffy/flash, other, is a form of prioritization. Computer hard disks hold the most data and are the most likely source of evidence. Next are PDAs and cell phones that may contain valuable contact information. The last is other storage devices. Special reminders, e.g., “Never leave evidence unattended,” are also indicated in the process flow diagram.

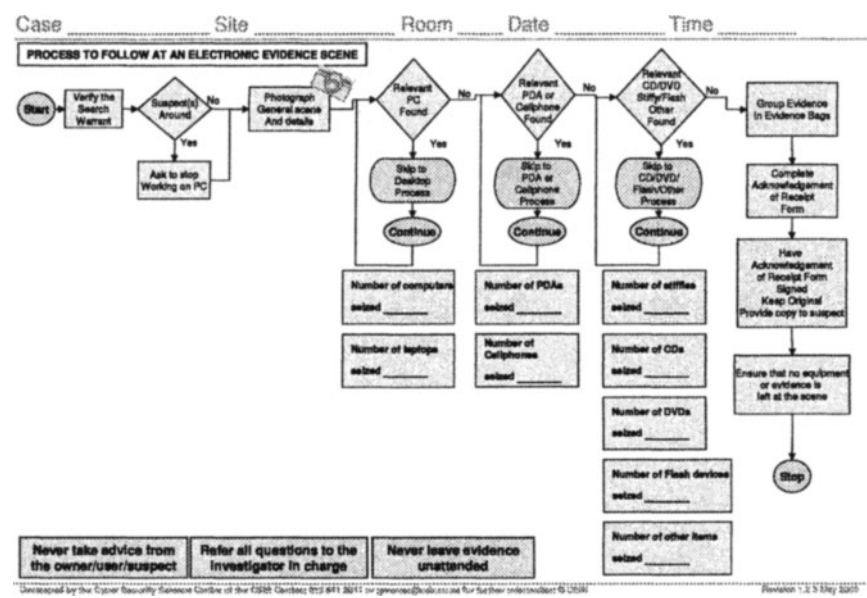


Figure 2. Process flow diagram for an electronic crime scene.

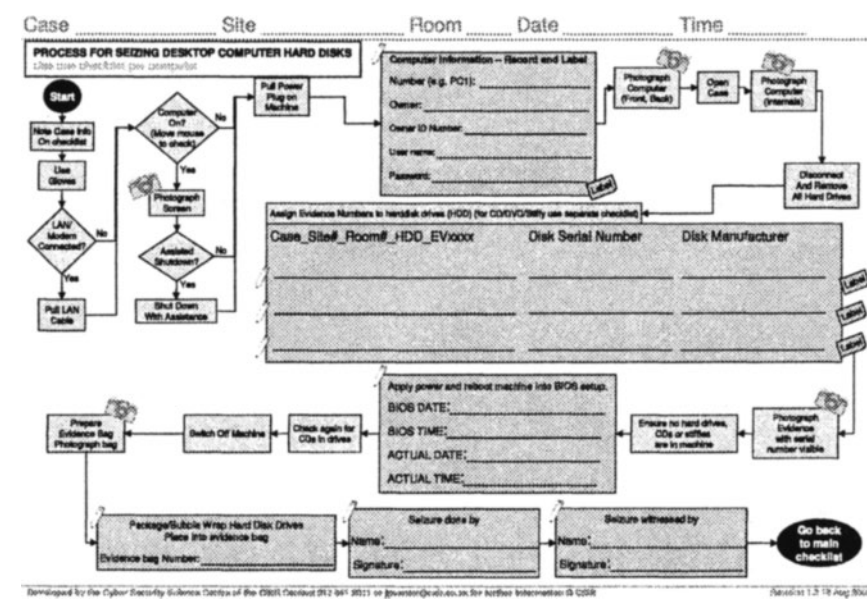


Figure 3. Process flow diagram for seizing desktop computer hard disks.

The process flow diagram for seizing desktop computer hard disks is shown in Figure 3. The first question that arose during its development was how to handle machines that are found to be running. Some experts recommend the immediate removal of the power cord from the machine [6], while others stress that evidence must be collected first [2]. The process flow in Figure 3 proposes an assisted shutdown: the normal operating system procedure is used to shut down the machine gracefully only if a technically competent person is available to assist. In the absence of technical support, the power cord should be removed from the machine. The rationale is that preserving the integrity of potential evidence on the hard disk is much more important than any volatile evidence that may be lost due to an immediate shutdown. Since it is important to tie the suspect to the machine [9], the owner's name and identifying number must be recorded.

Other tasks to be performed are photographing the front, back and insides of the computer, and noting the machine's BIOS date and time versus the actual date and time. This is necessary to connect file timestamps to the actual time of file access during the forensic analysis phase. The process flow diagram requires these tasks to be performed after all devices (e.g., hard disks, stiffies and CDs) are removed from the machine to ensure that potentially harmful programs are not triggered upon start up.

Figure 4 presents the process flow diagram for seizing PDAs and cell phones. These devices are grouped together due to their similar nature. It is important to obtain the PINs or passcodes of the devices as it is not possible—without much effort and cost—to obtain evidence without them. If the owner of a device is uncooperative, the investigator in charge of the scene must be notified. It is then up to the investigator to take further action. Note that the device must not be shut down if the PIN or passcode is not available.

PDAs and cell phones typically have to be recharged relatively soon (e.g., a week or two). Since the power supply and connector configurations are often unique to the devices, they must also be seized so that the devices can be recharged (if necessary) during the analysis phase [6].

The process flow diagram for seizing other non-volatile storage media is shown in Figure 5. Since a large number of these devices are often found at a crime scene, it may not be possible to label all of them at the scene. Therefore, the process flow diagram indicates that these items may be seized together, placed in an evidence bag, and labeled later.

Certain items suggested by other authors, e.g., BIOS passwords, encryption pass phrases [9], and the purpose of the system [6], are not recorded on any of the process flow diagrams. There are two reasons for

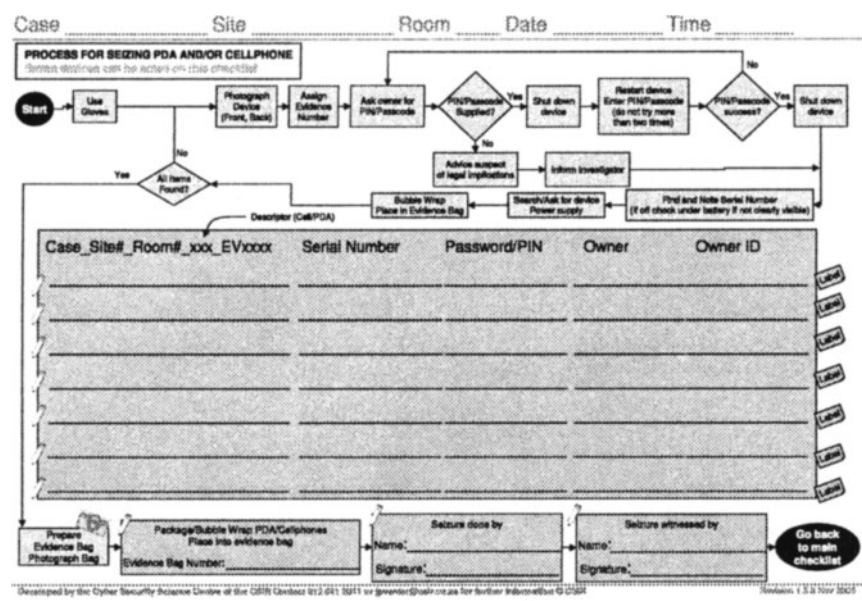


Figure 4. Process flow diagram for seizing PDAs and cell phones.

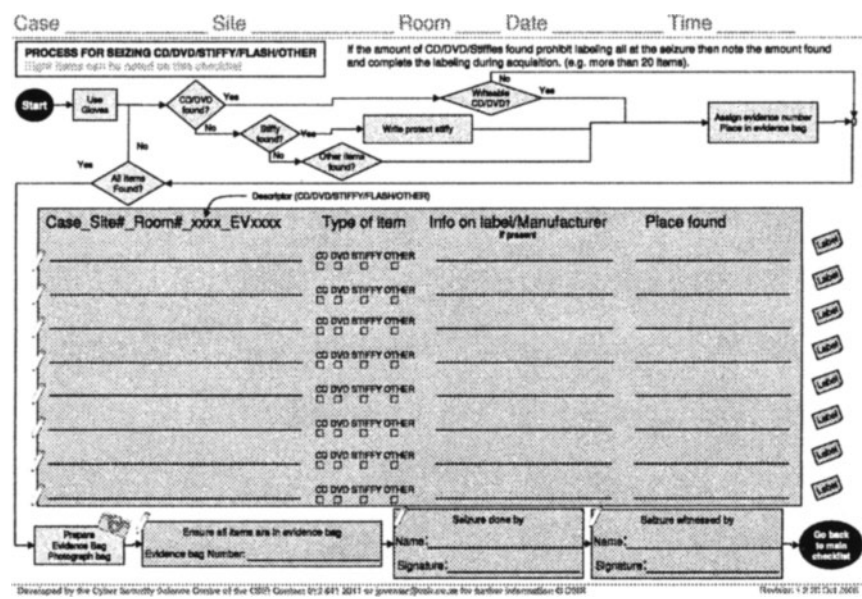


Figure 5. Process flow diagram for seizing non-volatile storage media.

this. The first is that the A4 layout does not provide enough space for all information to be recorded on the process flows. The second reason is that interactions between a first responder and a suspect should be minimized; this is to ensure that the suspect does not interfere with the process and possibly undermine the confidence of the first responder, which could lead to mistakes. We assume that the chief investigator—not the first responder—is responsible for interacting at length with suspects.

5. Impact of Process Flow Diagrams

A digital forensics course for first responders was developed and taught to four groups of law enforcement personnel in South Africa. The courses were presented by the South African Council for Scientific and Industrial Research from February through April 2005. Each group consisted of 15 participants and the courses were presented by three lecturers. The courses combined theory and practice with the specific aim of developing first responders who would have the skills and confidence to conduct searches and seizures involving digital evidence. Each participant was tested individually during a practical search and seizure session.

The process flow diagrams were developed during the first two training courses, and were tested on trainees during the third and fourth courses. The use of the process flow diagrams during the practical tests was lower than expected (53% total; 60% of those who passed). Based on the pass rates, it cannot be concluded that the process flows had a significant impact.

However, other observations made during the training courses provide better indicators of the impact of process flow diagrams. The first was a noticeable decrease in seizure times. In the first course, where process flows were not used, although most of the participants had previous experience, many still struggled to complete the seizure test within the one hour allocated to them. In the fourth course, where the process flow diagrams were used, most of the participants completed the seizure test in less than one hour. The quickest seizure time dropped from 55 minutes in the first course to 40 minutes in the fourth course.

In general, participants who used the process flow diagrams completed the seizures in less time, made fewer mistakes, and were more relaxed and confident. During the course feedback sessions, participants indicated that they preferred the process flow diagrams over traditional checklists, and mentioned they would use them in their operational activities.

6. Conclusions

Process flow diagrams are a powerful means for detailing the actions that first responders must perform when executing search and seizure warrants at electronic crime scenes. The evaluation of process flows indicates that are useful for training individuals with limited expertise in information and communications technology. Using process flow diagrams also contributed to increased confidence on the part of first responders and faster seizure times. Moreover, process flows have significant operational value, and often speed up investigations by experienced law enforcement personnel.

Several enhancements are possible. These range from incorporating a means to indicate photo numbers and other references in process flows to creating an acquisition process flow diagram, which details the actions to be performed during an on-site acquisition of evidence. Our basic process flow structure supports the development of new process flow diagrams while preserving the simplicity desired by first responders. Other enhancements include customizing process flow diagrams, and using them in conjunction with a computerized case management system. This would improve the quality and efficiency of searches and seizures, as well as every phase of the digital forensic process—from evidence acquisition and examination to analysis and courtroom presentation.

References

- [1] N. Beebe and J. Clark, A hierarchical, objectives-based framework for the digital investigation process, *Digital Investigation*, vol. 2(2), pp. 147-167, 2005.
- [2] D. Brezinky and T. Killalea, Guidelines for evidence collection and archiving, RFC 3227, The Internet Society, 2002.
- [3] Institute for Security Technology Studies, Law Enforcement Tools and Techniques for Investigating Cyber Attacks: A National Needs Assessment, Technical Report, Dartmouth College, Hanover, New Hampshire, 2002.
- [4] National Hi-Tech Crime Unit, High-tech crime: The impact on UK business—2005, London, United Kingdom, 2005.
- [5] M. Rogers and K. Seigfried, The future of computer forensics: A needs analysis survey, *Computers & Security*, vol. 23(1), pp. 12-16, 2004.
- [6] U.S. Department of Justice, *Electronic Crime Scene Investigation: A Guide for First Responders*, Washington, DC, 2001.

- [7] J. Vacca, *Computer Forensics: Computer Crime Scene Investigation*, Charles River Media, Hingham, Massachusetts, 2002.
- [8] H. Wolfe, Computer forensics, *Computers & Security*, vol. 22(1), pp. 26-28, 2003.
- [9] H. Wolfe, The circumstances of seizure, *Computers & Security*, vol. 22(2), pp. 96-98, 2003.
- [10] H. Wolfe, Forensics evidence testimony—Some thoughts, *Computers & Security*, vol. 22(7), pp. 577-579, 2003.
- [11] H. Wolfe, Setting up an electronic evidence forensics laboratory, *Computers & Security*, vol. 22(8), pp. 670-672, 2003.

Chapter 27

A CONTROL FRAMEWORK FOR DIGITAL FORENSICS

S. von Solms, C. Louwrens, C. Reekie and T. Grobler

Abstract This paper introduces a control framework for digital forensics. It proposes a taxonomy for control objectives, categorized within the phases of the digital forensic process: planning and preparation, incident response, investigation and juridical/evidentiary. Using the taxonomy as a basis, a digital forensic reference framework, consisting of control groupings, control objectives and detailed control objectives, is defined. The control framework is intended to provide a sound theoretical basis for digital forensics as well as a reference framework for digital forensics governance within organizations.

Keywords: Digital forensics control framework, control objectives, governance

1. Introduction

Digital forensics (a.k.a. computer forensics) is a relatively new discipline, which has not as yet been adequately defined within a governance framework comparable to COBIT (Control Objectives for Information and Related Technology) [6] or ITIL (IT Infrastructure Library) [7]. The importance of digital forensics and its implementation within organizations are not well understood by organizations. According to Michael Bacon, evangelist and principal of archolutions.com [1]:

“Computer forensics is an area where many companies fear to tread—until they have to. It requires specialist training, not only in technology, but also in evidence gathering and presentation in court. Few corporates are prepared to invest the time and money in their own staff to train them up.”

A proper digital forensics governance framework is needed to address these issues. As no formal framework currently exists, we attempt to define one based on the literature in digital forensics and our experience

Please use the following format when citing this chapter:

von Solms, S., Louwrens, C., Reekie, C., Grobler, T., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Shenoi, S., (Boston: Springer), pp. 343–355.

in the discipline. The framework described in this paper incorporates control objectives (COs) and detailed control objectives (DF-DCOs), and is intended to provide a sound theoretical basis for digital forensics as well as a reference framework for digital forensics governance within organizations.

This paper begins by presenting the problem statement, definitions and phases of the digital forensic process, which form the basis for defining the relevant digital forensic control groupings. Next, the elements identified in the presentation are discussed briefly in formulating the control objectives (COs) and detailed control objectives (DCOs) for each phase. The consolidated COs and DCOs comprise the digital forensic control framework.

2. Digital Forensic Process

Digital forensics can be defined in different ways. We adopt the following definition for the purposes of this work:

Digital forensics comprises analytical and investigative techniques used for the preservation, identification, extraction, documentation, analysis and interpretation of computer media, which are digitally stored or encoded for evidentiary and/or root cause analysis.

Casey [4] states that a standard operating procedure (SOP) should be performed whenever a computer is collected and/or examined. We call these SOPs “forensically-sound processes.” Such processes maintain the integrity of evidence, ensuring that the chain of custody remains unbroken and that the collected evidence will be admissible in a court of law.

According to Kruse and Heiser [9], the digital forensic process involves four activities: (i) securing the evidence without contaminating it, (ii) acquiring the evidence without altering or damaging the original, (iii) authenticating that the recovered evidence is the same as the original seized data, and (iv) analyzing the data without modifying it. On the other hand, Carrier [2] identifies three phases in crime scene investigations: (i) system preservation, (ii) evidence search, and (iii) event reconstruction.

Kruse and Heiser as well as Carrier see the first step in the forensic process as securing/preserving the evidence in response to an incident, which is clearly reactive in nature. Rowlingson [12] argues that considerable effort should be put into “forensic readiness” to serve as an enabler for the subsequent incident response and investigation phases. Thus, planning and preparation must also be emphasized in the forensic process.

Kruse and Heiser [9] stress that criminal prosecution is one of the major goals of digital forensics. We can, therefore, postulate that the digital forensic process must also include a “juridical” or evidentiary phase. The term juridical refers to judicial proceedings or relating to the law.

3. Digital Forensics Control Framework

In our view, the digital forensic process has four phases: (i) planning and preparation (readiness), (ii) incident response (evidence preservation), (iii) investigation (evidence acquisition, authentication, search and analysis), and (iv) juridical or evidentiary (event reconstruction, root cause analysis and evidence presentation). The terms in parentheses denote the high-level control objectives for each phase.

The following sections discuss the four phases of the digital forensic process and their associated control objectives in detail.

3.1 Planning and Preparation Phase

This section focuses on the planning and preparation phase. Four high-level digital forensic control objectives (COs) and twenty-one detailed control objectives (DCOs) are defined for this phase, which collectively form Group I: Digital Forensic Readiness (see Table 1).

Digital Forensic Readiness: Digital forensic readiness is the ability of an organization to maximize its potential to use digital evidence whilst minimizing the costs of an investigation. The goals of forensic readiness [12] are to: 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, minimize interruption to the business from any investigation, and ensure that the evidence makes a positive impact on the outcome of any legal action.

The following ten steps describe the key activities involved in implementing a forensic readiness program [12]:

- Define the business scenarios that require digital evidence.
- Identify available sources and different types of potential evidence.
- Determine the evidence collection requirement.
- Establish a capability for securely gathering legally admissible evidence to meet the requirement.

- Establish a policy for secure storage and handling of potential evidence.
- Ensure monitoring is targeted to detect and deter major incidents.
- Specify circumstances when escalation to a full formal investigation (which may use the digital evidence) should be launched.
- Train staff in incident awareness so that all those involved understand their role in the digital evidence process and the legal sensitivities of evidence.
- Document an evidence-based case describing the incident and its impact.
- Ensure legal review to facilitate action in response to the incident.

Computer Emergency Response Team: It is essential to establish a Computer Emergency Response Team (CERT) to ensure that the activities mentioned above are effectively utilized and executed following an incident. The CERT would have responsibilities and functions pertaining to planning and preparation, and incident response.

Policies Facilitating Investigations: Yasinsac and Manzano [12] note that enterprise policies can enhance computer and network forensics. They propose six categories of policies to facilitate digital forensic investigations: (i) retaining information, (ii) planning the response, (iii) training, (iv) accelerating the investigation, (v) preventing anonymous activities, and (vi) protecting the evidence.

The first four categories are included as high-level digital forensics control objectives in Group I: Digital Forensic Readiness (see Table 1). However, the first category (retaining information) is modified to “planning information retention requirements” to better describe the actions required during this phase. The remaining two categories (preventing anonymous activities and protecting the evidence) are included as detailed control objectives (DCOs). In all, Group I: Digital Forensic Readiness (Table 1) incorporates four high-level digital forensic control objectives (COs) and twenty-one detailed control objectives (DCOs).

3.2 Incident Response Phase

The incident response phase incorporates four high-level digital forensic control objectives (COs) and thirteen detailed control objectives (DCOs), which collectively form Group II: Evidence Preservation (see Table 2).

Table 1. Group I: Digital Forensic Readiness (4 COs with 21 DCOs).

DFR1 #	Planning Information Retention Requirements
DFR1.1	Define the business scenarios that require digital evidence.
DFR1.2	Identify available sources and different types of potential evidence.
DFR1.3	Determine the evidence collection requirement.
DFR1.4	Establish a policy for secure storage and handling of potential evidence.
DFR1.5	Establish a capability for securely gathering legally admissible evidence to meet the requirement.
DFR1.6	Synchronize all relevant devices and systems.
DFR1.7	Gather potential evidence.
DFR1.8	Prevent anonymous activities.
DFR2 #	Planning the Response
DFR2.1	Ensure monitoring is targeted to detect and deter major incidents.
DFR2.2	Implement intrusion detection systems.
DFR2.3	Specify circumstances when an escalation to a full formal investigation (which may involve digital evidence) should be launched.
DFR2.4	Establish a Computer Emergency Response Team (CERT).
DFR2.5	Establish capabilities and response times for external digital forensic investigation professionals.
DFR3 #	Digital Forensic Training
DFR3.1	Train staff in incident awareness, so that all understand their roles in the digital evidence process and the legal sensitivities of evidence.
DFR3.2	Develop an in-house investigative capability, if required.
DFR3.3	Enhance capability for evidence retrieval.
DFR4 #	Accelerating the Digital Forensic Investigation
DFR4.1	Document and validate an investigation protocol against best practice.
DFR4.2	Acquire appropriate digital forensic tools and systems.
DFR4.3	Ensure legal review to facilitate action in response to the incident.
DFR4.4	Define responsibilities and authority for CERT and investigative teams.
DFR4.5	Define circumstances for engaging professional investigative services.

Evidence Preservation: The purpose of this phase is to preserve the state of the crime scene. The first step in the incident response phase is to alert the CERT and initiate the incident response plan. One of the first tasks of the CERT is to secure all relevant evidence.

Evidence Transportation: The FBI's *Handbook of Forensic Services* [5] outlines a procedure for packing and shipping evidence, including computers and other electronic devices. A file should be stored

Table 2. Group II: Evidence Preservation (4 COs with 13 DCOs).

EPV1 #	Incident Response
EPV1.1	Initiate incident response plan.
EPV1.2	Activate the CERT.
EPV2 #	Secure Evidence
EPV2.1	Secure the physical environment of the crime scene.
EPV2.2	Secure all relevant logs and data.
EPV2.3	Secure volatile evidence, including laptops.
EPV2.4	Secure hardware.
EPV2.5	Label and seal all exhibits.
EPV2.6	Preserve chain of evidence.
EPV3 #	Transport Evidence
EPV3.1	Securely transport evidence.
EPV3.2	Preserve chain of custody during transport.
EPV4 #	Store Evidence
EPV4.1	Store evidence in safe custody room.
EPV4.2	Control access to evidence.
EPV4.3	Preserve chain of custody in storage.

on WORM (write-once-read-many) media, with a cryptographic hash stored offline in a physically secure container. Chain of custody documentation should be updated to reflect tracking numbers and other information [13]. Integrity of data that has undergone network transport may be proven via cryptographic hashing prior to sending and after receiving the data, and then comparing the hash values [13].

Evidence Storage: The objective of physical evidence storage is to provide a provable means of restricted access to evidence. Ultimately, a secure container in an audited access controlled room with camera monitoring and limited traffic would provide a foundation for the secure physical storage of evidence [13].

3.3 Investigation Phase

This section specifies five high-level digital forensic control objectives (COs) and eight detailed control objectives (DCOs) for the investigation

Table 3. Group III: Forensic Acquisition (5 COs with 8 DCOs).

FACQ1 #	Ensure Integrity of Evidence
FACQ1.1	Follow established digital forensic investigation protocols.
FACQ1.2	Write-protect all evidence source media.
FACQ2 #	Acquire Evidence
FACQ2.1	Acquire evidence in order of volatility.
FACQ2.2	Acquire non-volatile evidence.
FACQ3 #	Copy Evidence
FACQ3.1	Make forensic copies of all evidence.
FACQ4 #	Authenticate Evidence
FACQ4.1	Authenticate all evidence as identical to the original.
FACQ4.2	Time stamp all copies of the authenticated evidence.
FACQ5 #	Document Acquisition Process
FACQ5.1	Document all actions through chain of custody documentation.

phase, which collectively form Group III: Forensic Acquisition (see Table 3). The main aspects of the investigation phase are discussed below.

Forensic Acquisition: Forensic acquisition typically amounts to collecting volatile data (RAM, register state, network state, etc.) and imaging (forensic duplication) of disks. This process must use forensically-sound methods and conform with the widely-accepted order of volatility (OOV), which takes into account the fact that collecting some data affects other data.

Forensic Duplication: Forensic duplication of target media produces a “mirror image” of the target system. It also provides a working copy of the target media for analysis without the danger of altering or destroying potential evidence [11]. File-level copies, such as normal backups, do not yield all the potential evidence (e.g., deleted files, residual data on slack space and unallocated clusters).

Evidence Authentication: The integrity of the data/evidence must be unquestionable throughout preservation, acquisition, analysis and presentation. For this reason, the data should be cryptographically

hashed both collectively and individually, and the hashes themselves should be time-stamped.

Time-Stamping: According to Tan [13]: “Electronic documents will only stand up in court if the who, what and when they represent are unassailable.” Evidence presented in court must, therefore, be time-stamped whenever possible.

Chain of Evidence Preservation: It is of paramount importance that the chain of evidence remains unbroken to ensure that an intact causal chain exists. Casey [3] introduces a scale (Casey’s Certainty Scale) by which the trustworthiness of digital evidence can be assessed.

Chain of Custody: The objective of a chain of custody document is to track who had access to a given piece of evidence, when and, optionally, for what purpose. The life of a chain of custody document should start when the data is first considered as “potential evidence” and should continue through the presentation of the item as evidence in court [13].

3.4 Forensic Analysis Phase

Carrier [2] refers to forensic analysis as “evidence searching.” He identifies four key actions: surveying the available evidence, setting a hypothesis, search for data to support or refute the hypothesis, and documenting the findings.

We define six high-level digital forensic control objectives (COs) and fourteen detailed control objectives (DCOs) for this phase, which collectively form Group IV: Forensic Analysis (see Table 4). A discussion of the main aspects of the forensic analysis phase follows.

Investigation Planning: All relevant information regarding the incident needs to be reviewed to determine what expertise is required and which forensic tools would be the most appropriate to use.

Hypothesis Development: A set of hypotheses must be developed to cover the most likely scenarios. Next, a set of criteria should be developed to either prove or disprove a specific hypothesis.

Evidence Acquisition: The evidence should be acquired using the most suitable forensic tool. It is important to use tools that have a proven track record and will be acceptable in court.

Table 4. Group IV: Forensic Analysis (6 COs with 14 DCOs).

FAN1 #		Plan Investigation
FAN1.1	Review all available information regarding the incident.	
FAN1.2	Identify expertise required.	
FAN1.3	Identify most suitable tools to be utilized.	
FAN2 #		Develop Hypothesis
FAN2.1	Develop a hypothesis to cover most likely scenarios.	
FAN2.2	Define criteria to prove or disprove the hypothesis.	
FAN3 #		Acquire Evidence
FAN3.1	Acquire evidence using the most suitable tools available.	
FAN3.2	Analyze evidence using the most suitable tools available.	
FAN3.3	Conform to the requirements of the "best evidence rule."	
FAN4 #		Test Hypothesis
FAN4.1	Reconstruct sequence of events.	
FAN4.2	Compare evidence with other known facts.	
FAN5 #		Make Finding
FAN5.1	Make a finding that is consistent with all the evidence.	
FAN5.2	Document the finding.	
FAN6 #		Document Case
FAN6.1	Document all aspects of the case.	
FAN6.2	Enter documentation into safe custody.	

Best Evidence Rule: Courts sometimes require the original written material, recordings and photographs to exhibited as evidence [3]. This was intended to prevent witnesses from misrepresenting such materials and simply accepting the testimony regarding the contents. With the advent of photocopiers, scanners, computers and other technology that create effectively identical duplicates, copies became acceptable in place of the originals, unless a genuine question was raised about the authenticity of the original, the accuracy of the copy or if, under the circumstances, it would be unfair to admit the copy in lieu of the original. Because an exact duplicate of most forms of digital evidence can be made, a copy is generally acceptable. In fact, presenting a copy of digital evidence is usually more desirable because it eliminates the risk

that the original will be accidentally altered. However, this may vary according to the jurisdiction. Section 15 of the South African Electronic Communications and Transactions Act 25 of 2002 (ECT Act) provides that the rules of evidence must not be applied to deny the admissibility of a data message purely because it is constituted by a data message, or on the grounds that it is not in its original form, if it is the best evidence that the person adducing it can obtain.

Hypothesis Testing: The hypothesis must be tested against the acquired evidence and should enable investigators to reconstruct credible sequences of events. It should also be compared to other known facts [2].

Findings: Once all possible evidence has been considered, a finding can be made that is consistent with all the known facts. The finding—as well as the reasoning behind it—should be documented.

Documentation: The case must be thoroughly documented. This includes all the investigative actions taken, extracts of the evidence, deductions and findings. It is important to note that these facts may be called into question in litigation many years after the investigation.

3.5 Juridical/Evidentiary Phase

The juridical/evidentiary phase is arguably the most important phase in the digital forensic process as it determines if all the preceding effort will bear fruit or come to nothing. This phase involves three steps that must be taken to ensure a successful conclusion to the case: case preparation, case presentation and evidence preservation. Three high-level digital forensic control objectives (COs) and ten detailed control objectives (DCOs), corresponding to Group V: Evidence Presentation, are defined for this phase (see Table 5).

Case Preparation: Legal requirements vary according to jurisdiction. Due to the nature of cyber crime, many investigations will involve international components and may require investigators to conform to several differing legal requirements.

Expert witnesses must be identified and thoroughly prepared. Exhibits must also be prepared taking the target audience into consideration. These can include presentation aids like graphics, slide shows, photographs, hardware and even live demonstrations.

Table 5. Group V: Evidence Presentation (3 COs with 10 DCOs).

EP1 #	Prepare Case
EP1.1	Determine target audience (court, disciplinary hearing, inquiry).
EP1.2	Assemble evidence required for presentation.
EP1.3	Prepare expert witnesses.
EP1.4	Prepare exhibits.
EP1.5	Prepare presentation aids (graphics, slides, hardware).
EP1.6	Preserve chain of custody.
EP2 #	Present Case
EP2.1	Present evidence in a logical, understandable way to ensure that the court can critically assess every bit of information and understand the relevance to the case.
EP2.2	Make use of graphics and physical examples to illustrate difficult or critical concepts, if needed.
EP2.3	Ensure that a digital forensic specialist is at hand to assist in providing expert evidence.
EP3 #	Preserve Evidence
EP3.1	Preserve the evidence after the case has been presented, as it may be needed in case of appeal or if new evidence is obtained.

Vacca [14] lists four tests that should be applied to evidence: (i) authenticity, (ii) reliability, (iii) completeness, and (iv) freedom from interference and contamination.

Case Presentation: Evidence must be presented in a logical, understandable way to ensure that the court can critically assess every bit of information and understand its relevance to the case.

Since the investigative process can be attacked by the defense, regardless of how overwhelming the evidence might be, specific care should be taken to stress that an internationally accepted forensic process had been followed and that the chains of evidence and custody have remained intact throughout the process.

Evidence Preservation: Evidence must be preserved securely after the case has been presented as it may be needed again in case of appeal or if new evidence becomes known.

4. Conclusions

The digital forensics reference framework presented in this paper incorporates five high-level digital forensics control objectives: (i) digital forensic readiness, (ii) evidence preservation, (iii) forensic acquisition, (iv) forensic analysis, and (v) evidence presentation. These five digital forensic control groupings are refined into 22 control objectives (COs), which are further refined into 66 detailed control objectives (DCOs). Several of the DCOs relate to “forensically-sound processes,” which must be executed in sequence or in conjunction with each other. The reference framework is intended to provide a sound theoretical basis for digital forensics as well as a foundation for the practical implementation of digital forensics governance within organizations.

We are currently engaged in mapping the digital forensics control objectives (COs) against well-established governance frameworks like COBIT, ISO/IEC 17799 [8] and ITIL [7]. Interested readers are referred to [10] for preliminary results of this effort.

References

- [1] I. Armstrong, Computer forensics: Detecting the imprint, *SC Magazine*, August 2002.
- [2] B. Carrier, *File System Forensic Analysis*, Addison-Wesley, Upper Saddle River, New Jersey, 2005.
- [3] E. Casey, *Digital Evidence and Computer Crime*, Academic Press, London, United Kingdom, 2001.
- [4] E. Casey (Ed.), *Handbook of Computer Crime Investigation: Forensic Tools and Technology*, Elsevier Academic Press, London, United Kingdom, 2004.
- [5] Federal Bureau of Investigation, *Handbook of Forensic Services* (www.fbi.gov), 2003.
- [6] Information Technology Governance Institute, *COBIT: Control Objectives for Information and Related Technologies*, Rolling Meadows, Illinois, 2000.
- [7] Information Technology Infrastructure Library (www.itil.co.uk), Office of Government Commerce, London, United Kingdom.
- [8] International Organization for Standardization, *ISO/IEC 17799: Code of Practice for Information Security Management*, Geneva, Switzerland, 2000.
- [9] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*, Addison-Wesley, Reading, Massachusetts, 2002.

- [10] C. Louwrens and S. von Solms, The relationship between digital forensics, corporate governance, information technology governance and information security governance, in *Digital Crime and Forensic Science in Cyberspace*, P. Kanellis, E. Kiountouzis, N. Kolokotronis and D. Martakos (Eds.), Idea Group, Hershey, Pennsylvania, 2006.
- [11] K. Mandia, C. Proise and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.
- [12] R. Rowlingson, A ten step process for forensic readiness, *International Journal of Digital Evidence*, vol. 2(3), pp. 1-28, 2004.
- [13] J. Tan, Forensic readiness (www.webproxy.com/research/reports/acrobat/atstake_forensic_readiness.pdf), July 17, 2001.
- [14] J. Vacca, *Computer Forensics: Computer Crime Scene Investigation*, Charles River Media, Hingham, Massachusetts, 2002.

Chapter 28

CRIMINAL REGULATION OF ANTI-FORENSIC TOOLS IN JAPAN

Tetsuya Ishii

Abstract This paper discusses the continuing landmark debate in a Japanese Court concerning the development and distribution of a peer-to-peer (P2P) file sharing program. The program, known as Winny, facilitates illegal activities such as piracy and the distribution of child pornography because of the encryption and anonymity afforded to users. The court has to determine whether Isamu Kaneko, the designer of Winny, is criminally liable for developing and distributing the program. This paper also assesses whether the judgment in the Winny case might set a precedent for regulating the creation and distribution of anti-forensic tools.

Keywords: Anti-forensic tools, peer-to-peer networks, legal issues

1. Introduction

Digital forensics is the application of computer investigation and analysis techniques to gather and assess digital evidence for use in legal proceedings. By encrypting and eliminating data before it can be seized, so-called “anti-forensic tools” hinder the course of justice and prevent the successful prosecution of criminals. As a result, these tools can be seen to be encouraging the perpetration of digital crimes. The key issue, therefore, is whether or not the development and distribution of anti-forensic tools should be made illegal.

Articles 2 through 5 of the Council of Europe’s Convention on Cyber Crime (2001) proscribe offenses against the confidentiality, integrity and availability of computer data. Article 6 prohibits using a tool to commit such an offense, but the Convention does not specifically address the development and distribution of anti-forensic tools. The Convention states that it is permissible to punish the user of a tool if it is used to

Please use the following format when citing this chapter:

Ishii, T., 2006 in International Federation for Information Processing, Volume 222, Advances in Digital Forensics II, eds. Olivier, M., Sheno, S., (Boston: Springer), pp. 357–364.

commit an illegal activity like erasing evidence. But the Winny case requires the court to decide whether such recourse can be extended to developing and distributing such a tool over the Internet.

This paper presents the legal issues being debated in the case. Also, it evaluates the prospect of regulations being introduced in Japan that proscribe the use of anti-forensic tools.

2. The Winny Case

On May 10, 2004, Isamu Kaneko, a research associate at the Graduate School of Information Science and Technology, University of Tokyo, was arrested. Kaneko had developed and distributed the peer-to-peer (P2P) file sharing program, Winny, on the Internet. Many technical experts and analysts raised concerns about the effect his arrest could have on the right to Internet privacy and the future development of P2P technology. Kaneko was charged with aiding and abetting the sharing of copyrighted material by two Japanese users of Winny, a 19-year-old man in Matsuyama and 41-year-old man in Gumma. The former used Winny to illegally share copies of gaming software; the later illegally shared copies of movies. The Kyoto District Court [4] found that the two principals had violated the public transmitting rights of the copyright holders. They were charged and found guilty under clauses 96bis and 119, Article 23, of the Japanese Copyright Act:

- Article 23: Rights of Public Transmission
 - (i) The author shall have the exclusive right to publicly transmit his work (including the “making transmittable” of his work, in the case of interactive transmission media).
 - (ii) The author shall have the exclusive right to communicate publicly, by means of a receiving apparatus, his work from which the public transmission has been made.
- Article 96bis: Right of Making Transmittable

Producers of phonograms shall have the exclusive right to make their phonograms transmittable.
- Article 119: Right of Making Transmittable

The following shall be punishable by imprisonment for a term not exceeding five years or a fine not exceeding five million Yen, or both:

 - (i) Any person who infringes the moral rights of authors, copyright, right of publication, moral rights of performers or neighboring rights (excluding those who reproduce by themselves

works or performances, etc. for the purpose of private use as mentioned in Article 30, paragraph 1 (including the case where its application *mutatis mutandis* is provided for under the provision of Article 102, paragraph 1), those who do an act considered to constitute an infringement on the moral rights of authors, copyright, moral rights of performers or neighboring rights (including the rights considered as neighboring rights in accordance with the provisions of Article 113, paragraph 4; the same shall apply in Article 120bis, item (iii)) under the provisions of Article 113, paragraph 3, or those who do an act considered to constitute an infringement on copyright or neighboring rights under the provisions of Article 113, paragraph 5.

- (ii) Any person who, for profit-making purposes, causes others to use automatic reproducing machines mentioned in Article 30, paragraph 1, item (i) for such reproduction of works or performances, etc. as constitutes an infringement on copyright, right of publication or neighboring rights.

In the following, we discuss relevant details about the Winny system and its development [3].

Kaneko developed Winny because he was inspired by “Share,” a P2P software program that is server-independent. Winny builds a virtual proxy cache server, which allows its users to download large files (e.g., movies or games) in an anonymous manner. Furthermore, Winny caches many files when it is activated, but the file names and the contents of the cached files are encrypted, which further supports anonymity. Consequently, Winny users are unaware if the contents of their files are lawful or not; this feature facilitates the sharing of unlawful files. Moreover, because Winny builds a virtual cache server and stores the encrypted files in it, it is not possible to delete a file once it is uploaded or to trace the individual who originally uploaded the file. Even if the original uploader deletes the original file from his hard drive, a copy of the file remains on the Winny system.

The development and the user acceptance testing of Winny was discussed on *2 channel*, a popular bulletin board. The defendant, Kaneko, enthusiastically participated in the discussion of the thread, “What comes after MX,” and he developed Winny in compliance with the wishes of other thread discussants who wanted to share unlawful files. MX refers to the WinMX file sharing program and “WinNY” was named to illustrate its evolutionary advancement (the letter N follows M, and Y follows X). At the time, WinMX users sharing illegal material and copyrighted

files were worried about being prosecuted because the authorities could easily detect who had uploaded the original files onto the WinMX network. Several users had, in fact, already been charged, and illegal file sharers were very keen to achieve complete anonymity. This became the acknowledged purpose of developing Winny. It seems highly unlikely, given the discussion on the bulletin board, that Kaneko did not know that the anonymity provided by Winny would aid and abet unlawful file sharing.

3. Legal Issues

This section discusses key legal issues related to Japanese criminal law.

3.1 Aiding and Abetting

Articles 62 and 63 of the Japanese Penal Code [1], which were established in 1907, state:

- Article 62: Accessory
 - (1) Any person who aids and abets a principal is an accessory.
 - (2) Any solicitant of an accessory shall be penalized as an accessory.
- Article 63: Penalties for Accessory

Any person who aids and abets shall be penalized by mitigating the one for the principal.

Under Japanese law, “aiding” means to offer physical assistance, while “abetting” means to offer mental assistance. Aiding and/or abetting are deemed to have occurred when such actions assist a principal to carry out a criminal activity. It is also necessary that the assisting party’s actions are found to be intentional.

To be deemed an accessory, a causal relationship between the conduct of the principal and the assistance of the accessory must be shown. According to commonly accepted theory, the causal relationship for an accessory is not the level of *conditio sine qua non*, which is required for the principal. Finding that the principal’s conduct was aided physically and/or mentally by the accessory satisfies the causal relationship requirement. Whether the accessory’s action made the conduct of the principal easier is the decisive factor. For example, mental assistance could include an accessory telling the principal that he will take care of the principal’s family while he is in prison. Physical assistance could include an accessory giving the principal a gun for a bank robbery, even though the principal used his own gun to commit the crime. Although the principal could have committed the robbery without the borrowed

gun, the gun's owner still became an accessory the moment the principal decided to commit the robbery because the accessory's assistance is likely to have encouraged his course of action.

The "degree of strengthening" is also requirement. If an accessory lent a cap to a murderer, for example, the assistance provided would not be strong enough to be considered abetting. Legal precedents suggest that a causal relationship exists when the criminal conduct of the principal and/or its result would not have been assisted without the accessory's action. Court judgments of "cause and effect of an assistance" have relied on the same interpretation.

Based on our understanding of Japanese law, should the court adjudge the development and distribution of Winny as aiding and abetting piracy? One aspect of this case makes the issue more complicated. The prosecutor argued that, not only did the development and distribution of Winny constitute a crime, but the user acceptance testing also constituted a crime because the 2 *channel* users clearly wanted to share unlawful files anonymously, without fear of arrest. Indeed, the defendant enhanced the anonymity provided by Winny precisely for this purpose. Winny aids piracy "physically" by making anonymity possible, and it abets the conduct of principals "mentally" by ensuring anonymity. Therefore, the court is likely to hold that the development and the distribution of Winny aided and abetted the principals' conduct.

3.2 Accessory with Neutral Conduct

There is also the issue of being an accessory, but with "neutral," "normal" or "usual" conduct in both Japanese law and its mother law (German criminal law). The issue is whether the development and distribution of Winny should be categorized as assistance because what Winny performs "is an action of neutral position." One analyst observed that Winny is just a file sharing system—a tool of neutral position; consequently, the distribution of a neutral tool does not constitute a crime. The neutral position argument holds that a tool is neutral until appropriated by an agent. For example, even if a knife was used to commit murder, the knife manufacturer and seller should not be considered to be accessories to murder. However, this is a circular argument. In reality, it merely helps to identify certain situations within which a tool could be categorized as "neutral." In other words, denying the assistance lent by a tool requires the positive refutation of either of the requirements applied to the assistance.

According to commonly accepted criminal law theory [6], the finding of accessory depends on whether a tool furnished the principal with a

particular function or capability with reference to the specific offense. The following explanation, presented by Jakobs [2], has been well established and is a commonly accepted interpretation. Jakobs uses the case of a baker who sells bread to a customer who plans to use the bread to poison someone. According to Jakobs, by selling "normal" bread to the customer, the baker did not participate in the murder perpetrated by the customer; accordingly, the baker is not an accessory for selling the bread. However, if the baker knowingly sold "special" bread, which helped the customer to conceal poison, then the baker should be found to be an accessory to the murder.

Nishida [5] explains:

"A cooking material shopkeeper sold a knife to his customer. He was afraid that the customer might commit a murder using the knife and the tragedy actually happened. However, the anxiety of the shopkeeper remains a vague uncertainty; therefore, he should not be found to be an accessory. On the contrary, if two people were fighting in front of the shop, and one ran into the shop to buy a knife and used it to kill the other, then the shopkeeper could be found to be an accessory."

Therefore, while there is a freedom to sell bread, the freedom to sell special bread that makes poisoning easy is not guaranteed. This should apply equally to the development of novel computer and network systems. There should definitely be the freedom to make new systems, but not systems that make specific crimes easier to commit.

It is important to attempt to establish an objective standard for deciding whether or not an individual is guilty of being an accessory. Although a comprehensive examination of this complex legal issue is beyond the scope of this paper, we can apply the basic legal position stated above: if a person knows it is highly probable that his/her conduct will assist the principal in the commission of a specific crime, that person can be considered an accessory to the crime.

In the case of Winny, the file sharing program physically aided the principals and mentally abetted them. The program's file sharing capabilities enabled the principals to physically commit their crimes. Its encryption and guaranteed anonymity encouraged the principals mentally because these features eliminated their fear of arrest.

Another, broader, interpretation can be made using the following legal principle: If the conduct of the accessory had not been made, would the principals have committed the offense? In the Winny case, it is reasonable to suggest that the principals would not have violated copyright law had Winny not provided a safe means to do so. Also, Kaneko's action of distributing Winny free of charge was more directly connected to the crime than just providing a cap to a principal before he or she committed the crime.

3.3 *Mens Rea*: Mental Requirement

Another interesting feature in the Winny case is that Kaneko, the defendant and designer of Winny, did not know the two principals. Normally, a person deemed to be an accessory knows the principal and, in most cases, the unlawful action the principal intended to commit; however, Kaneko did not. While Kaneko might have expected someone would commit the criminal act of piracy, he merely posted his program on the Internet as free software. In this situation, can we identify Kaneko's *mens rea* to charge him as an accessory to the principals' acts of piracy?

Under Japanese criminal law, it is enough that a person is aware of the range of possible danger his or her actions could create. Drawing attention to Kaneko's participation in the "What comes after WinMX" discussions on *2 channel*, and his expressed intentions to create the software, the prosecutor was able to argue that the principals' acts of piracy should be considered as being within the range. Therefore, Kaneko's development and distribution of Winny was a deliberate, intentional act.

4. Prospects for Anti-Forensic Tools

Winny incorporates excellent file sharing and encryption technologies; the resulting anonymity enables criminals to share unlawful data with little fear of arrest. Moreover, once a file is uploaded to Winny's virtual server for sharing, it is not possible to delete the file. In this sense, Winny is like Janus, the Roman god with two faces, one looking forward and one behind. Winny's front face demonstrates technological artistry, but the one behind is a tool for assisting criminal activity.

Some individuals argue that the successful prosecution of Winny's developer will stifle technological advances. They suggest that the steady evolution of Internet technology requires an environment of freedom, where researchers can explore uncharted territory without fear of reprisal or liability. However, such an approach is likely to prove more destructive and repressive in the long run. Reckless development can result in increasingly draconian legislative measures, resulting in a worse, highly policed environment for technological development.

This analysis applies across the board, and specifically covers malicious programs such as worms, viruses and spam. These programs are costly and destructive, and exemplify why technology should be bound by the same legal and ethical frameworks as other societal endeavors. At this time, the Japanese penal code has no explicit statutory provisions for tools capable of destroying digital evidence. However, the general principles of Japanese criminal law would indicate that the development

and distribution of anti-forensic tools should be regulated using the laws that proscribe the aiding and abetting of criminal activity. If the Winny case is applied to anti-forensic tools, in general, it should encourage developers to restrict and monitor the use of their tools and to attempt to ensure that they are not used with criminal intent.

5. Conclusions

The first public trial session in the Winny case was held on September 1, 2004. Due to the complexity of the case, the trial is still continuing (the 21st session ended on March 20, 2006). However, the Kyoto District Court can be expected to draw similar conclusions as those outlined in this paper. Kaneko's offense is that he was fully aware of the risk Winny presented, but did not take any measures to counter or reduce the danger before distributing Winny. This is likely to set the legal precedent that any designer of a computer program that could be used for criminal purposes, or a designer who is aware that the program is at risk of being used in a crime, should control the program and prevent it from being used for criminal purposes. Failure to do so could result in the designer of the program being charged as an accessory to the crime.

References

- [1] S. Dando, *The Criminal Law of Japan: The General Part*, B. George (Translator), Fred Rothman and Company, Littleton, Colorado, 1997.
- [2] G. Jakobs, Regressverbot beim Erfolgsdelikt, *Zeitschrift für Strafrechtswissenschaft*, vol. 89, pp. 1-35, 1977.
- [3] I. Kaneko, *The Technology of Winny*, ASCII, 2005.
- [4] Kyoto District Court Decision on November 30, Hanrei-Jihou No. 1858, p. 281, 2004.
- [5] N. Nishida and K. Souron, *The General Part of Criminal Law*, Kou-bundo, 2006.
- [6] S. Shimada, Kougi no Kyouhan no Ippannteki Seiritsu Youken (General Requirements of Accessories), *St. Paul's Review of Law and Politics*, vol. 57, pp. 44-153, 2001.