Advances in Digital Forensics

Edited by
Mark Pollitt
Sujeet Shenoi

Springer

ifip

Job #: 94346

Author name: Pollitt

Title of book: Advances in Digital Forensics

ISBN number: 0387300120

ADVANCES IN DIGITAL FORENSICS

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

IFIP International Conference on Digital Forensics, National Center for Forensic Science, Orlando, Florida, February 13-16, 2005

Edited by

Mark Pollitt
Digital Evidence Professional Services, Ellicott City, Maryland, USA

Sujeet Shenoi
University of Tulsa, Tulsa, Oklahoma, USA

Springer

Library of Congress Control Number: 2005934798

Advances in Digital Forensics
Edited by Mark Pollitt and Sujeet Shenoi

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

ISSN: 1571-5736 / 1861-2288 (Internet)
ISBN-10: 0-387-30012-0
ISBN-13: 9780-387-30012-0
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, Inc., 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.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 11580492
springeronline.com

Contents

Contributing Authors ix

Preface xvii

PART I THEMES AND ISSUES

1
Dealing with Terabyte Data Sets in Digital Investigations 3
Nicole Beebe and Jan Clark

2
Forensics and Privacy-Enhancing Technologies 17
Martin Olivier

3
A Network-Based Architecture for Storing Digital Evidence 33
Mark Davis, Gavin Manes and Sujeet Shenoi

4
Digital Forensics: Meeting the Challenges of Scientific Evidence 43
Matthew Meyers and Marcus Rogers

5
Non-Technical Manipulation of Digital Data 51
Michael Losavio

PART II INVESTIGATIVE TECHNIQUES

6
Detecting Social Engineering 67
Michael Hoeschele and Marcus Rogers

7
A Framework for Email Investigations 79
Anthony Persaud and Yong Guan

8
The Mitnick Case: How Bayes Could Have Helped 91
Thomas Duval, Bernard Jouga and Laurent Roger

9
Applying Forensic Principles to Computer-Based Assessment 105
R. Laubscher, D. Rabe, M. Olivier, J. Eloff and H. Venter

10
Exploring Forensic Data with Self-Organizing Maps 113
B. Fei, J. Eloff, H. Venter and M. Olivier

PART III NETWORK FORENSICS

11
Integrating Digital Forensics in Network Infrastructures 127
Kulesh Shanmugasundaram, Hervé Brönnimann and Nasir Memon

12
Using Peer-to-Peer Technology for Network Forensics 141
Scott Redding

13
Forensic Profiling System 153
P. Kahai, M. Srinivasan, K. Namuduri and R. Pendse

14
Global Internet Routing Forensics 165
Eunjong Kim, Dan Massey and Indrajit Ray

15
Using Signaling Information in Telecom Network Forensics 177
T. Moore, A. Meehan, G. Manes and S. Shenoi

PART IV PORTABLE ELECTRONIC DEVICE FORENSICS

16
Forensic Analysis of Mobile Phone Internal Memory 191
Svein Willassen

17
Imaging and Analysis of GSM SIM Cards 205
Christopher Swenson, Gavin Manes and Sujeet Shenoi

18
Extracting Concealed Data from BIOS Chips 217
P. Gershteyn, M. Davis, G. Manes and S. Shenoi

PART V LINUX AND FILE SYSTEM FORENSICS

19
Recovering Digital Evidence from Linux Systems 233
Philip Craiger

20
Detecting Hidden Data in Ext2/Ext3 File Systems 245
S. Piper, M. Davis, G. Manes and S. Shenoi

PART VI APPLICATIONS AND TECHNIQUES

21
Forensic Analysis of Digital Image Tampering 259
Gilbert Peterson

22
Content-Based Image Retrieval for Digital Forensics 271
Y. Chen, V. Roussev, G. Richard III and Y. Gao

23
Making Decisions about Legal Responses to Cyber Attacks 283
L. Peng, T. Wingfield, D. Wijesekera, E. Frye, R. Jackson and J. Michael

24
Applying Filter Clusters to Reduce Search State Space 295
Jill Slay and Kris Jorgensen

25
In-Kernel Cryptographic Executable Verification 303
Yusuf Motara and Barry Irwin

Contributing Authors

Nicole Beebe is a Ph.D. student in Information Technology at the University of Texas at San Antonio, San Antonio, Texas. Her research interests include digital forensics, information security, data warehousing and data mining.

Hervé Brönnimann is an Assistant Professor of Computer Science at Polytechnic University, Brooklyn, New York. His research interests include algorithms and computational geometry, data mining and data reduction, programming software libraries and network forensics.

Yixin Chen is an Assistant Professor of Computer Science at the University of New Orleans, New Orleans, Louisiana. His research interests include computer vision, machine learning and biomedical informatics.

Jan Clark is a Professor of Information Systems and Technology Management at the University of Texas at San Antonio, San Antonio, Texas. Her research interests include telecommunications, information security, information systems strategy and technology innovation.

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/computer security.

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.

Thomas Duval is a Computer Science Researcher with the French Ministry of Defense (DGA/CELAR), and a Ph.D. student in Computer Science at the University of Supélec, Rennes, France. His research interests include digital forensics, in particular, data recovery and correlation.

Jan Eloff is Professor and Head of the Department of Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include access control, risk analysis, risk management and web services 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.

Emily Frye is a Senior Legal Research Associate with the George Mason University School of Law's Critical Infrastructure Protection Program in Arlington, Virginia. Her research interests are in the area of critical infrastructure protection.

Yun Gao is a Ph.D. student in Computer Science at the University of New Orleans, New Orleans, Louisiana. His research interests include information security and digital forensics.

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

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

Michael Hoeschele is an M.S. student in Computer and Information Technology at Purdue University, West Lafayette, Indiana. His research interests include social engineering, digital forensics and ethics.

Barry Irwin is a Senior Lecturer in Computer Science at Rhodes University, Grahamstown, South Africa. His research interests include network security, Internet malware propagation and digital forensics.

Randall Jackson is a Senior Legal Research Associate with the George Mason University School of Law's Critical Infrastructure Protection Program in Arlington, Virginia. His research interests are in the areas of law and economics.

Kris Jorgensen is a Software Engineering graduate from the University of South Australia, Adelaide, Australia. He currently works for an Adelaide-based company.

Bernard Jouga is a Professor of Computer Science and Associate Director for Research and Industry Partnerships at the University of Supélec, Rennes, France. His research interests include telecommunications network protocols and information systems security.

Pallavi Kahai is a graduate student in Electrical and Computer Engineering at Wichita State University, Wichita, Kansas. Her research interests include statistical analysis, information security and computer forensics.

Eunjong Kim is an M.S. student in Computer Science at Colorado State University, Fort Collins, Colorado. Her primary research area is network security.

Rut Laubscher is Ph.D. student in Computer Science at the University of Pretoria and a Lecturer in Computer Information Systems at the Military Academy in Saldanha, South Africa. Her research interests include digital forensics and information warfare.

Michael Losavio, a Kentucky attorney, teaches in the Department of Computer Engineering and Computer Science at the University of Louisville, Louisville, Kentucky. His research interests include information security, digital forensics and the social impact of computing and information networks.

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 and telecommunications security.

Dan Massey is an Assistant Professor of Computer Science at Colorado State University, Fort Collins, Colorado. His primary research area is network infrastructure security.

Anthony Meehan is a graduate student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include converged network security and digital forensics.

Nasir Memon is an Associate 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.

Matthew Meyers is an M.S. student in Information Assurance and Security at Purdue University, West Lafayette, Indiana. His primary research area is digital forensics.

James Bret Michael is an Associate Professor of Computer Science, and Electrical and Computer Engineering at the Naval Postgraduate School, Monterey, California. His research interests include distributed computing and software engineering, with an emphasis on building highly dependable and trustworthy 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.

Yusuf Motara is an M.Sc. student in Computer Science at Rhodes University, Grahamstown, South Africa. His current research focuses on file integrity checkers as a method for improving security.

Kamesh Namuduri is an Assistant Professor of Electrical and Computer Engineering at Wichita State University, Wichita, Kansas. His research interests include wireless networks, video communications, information security and computer forensics.

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.

Ravi Pendse is an Associate Professor of Electrical and Computer Engineering and Associate Vice President of Academic Affairs at Wichita State University, Wichita, Kansas. His research interests include wireless networks, VoIP and storage area networks.

Leisheng Peng is a Ph.D. student in Information Technology at George Mason University, Fairfax, Virginia, and a senior programmer with System Integration, Inc. Her research interests include web-based systems, information security and automating legal frameworks.

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, machine learning and parallel processing.

Scott Piper is an undergraduate student in Computer Science at the University of Tulsa, Tulsa, Oklahoma. His research interests include digital forensics, network security and software vulnerability analysis.

Cobus Rabe is a Lecturer in Computer Information Systems at the Military Academy in Saldanha, South Africa. His research interests include digital forensics, information warfare and computer-based education.

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 computer security and digital forensics.

Scott Redding is a Ph.D. student in Computer Science at the University of Tulsa, Tulsa, Oklahoma, and Chief of the Information Technology Development Division at the U.S. Army Aberdeen Test Center, Aberdeen Proving Ground, Maryland. His research interests include information assurance and network security.

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.

Laurent Roger is a Security Researcher with the French Ministry of Defense (DGA/CELAR). His research interests include threat analysis and digital forensics.

Marcus Rogers is an Associate Professor of Computer and Information Technology at Purdue University, West Lafayette Indiana. His research interests include applied digital forensics, psychological crime scene analysis and information assurance.

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.

Kulesh Shanmugasundaram is a Ph.D. student in Computer Science at Polytechnic University, Brooklyn, New York. His research interests include network security, network monitoring and digital forensics.

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.

Jill Slay is the Director of the Enterprise Security Management Laboratory at the University of South Australia, Adelaide, Australia. Her research interests include information assurance, digital forensics, critical infrastructure protection and complex system modeling.

Mannivannan Srinivasan is a graduate student in electrical and computer engineering at Wichita State University, Wichita, Kansas. His research interests include VoIP, wireless networks and information 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 the Department of Computer Science, University of Pretoria, Pretoria, South Africa. His research interests include network security, digital forensics and intrusion detection systems.

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

Svein Willassen is a Ph.D. student in Telematics at the Norwegian University of Science and Technology, Trondheim, Norway. His research interests include digital forensics, mobile communications and data recovery.

Thomas Wingfield is a Research Fellow with the Potomac Institute for Policy Studies in Arlington, Virginia, and a Lecturer at the Catholic University of America's Columbus School of Law. His research interests are in the legal aspects of cyber security.

Preface

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

This book, *Advances in Digital Forensics*, is the first volume of a new 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 describes original research results and innovative applications in digital forensics. In addition, it highlights some of the major technical and legal issues related to digital evidence and electronic crime investigations.

This book contains twenty-five edited papers from the First Annual IFIP WG 11.9 Conference on Digital Forensics, held at the National Center for Forensic Science, Orlando, Florida, February 13-16, 2005. The papers were selected from more than forty-five submissions reviewed by members of IFIP Working Group 11.9.

The chapters are organized into six sections: themes and issues in digital forensics, investigative techniques, network forensics, portable electronic device forensics, Linux and file system forensics, and digital forensic applications and techniques. The coverage of topics highlights the richness and vitality of the new 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 acknowledge the institutional support provided by the National Center for Forensic Science and the University of Central Florida, Orlando, Florida. Most of all, we are grateful to the Federal Bureau of Investigation, the Internal Revenue Service and the National Security Agency for their longstanding support of the discipline of digital forensics and, in particular, the activities and initiatives of IFIP Working Group 11.9.

MARK POLLITT AND SUJEET SHENOI

I

THEMES AND ISSUES

Chapter 1

DEALING WITH TERABYTE DATA SETS IN DIGITAL INVESTIGATIONS

Nicole Beebe and Jan Clark

Abstract Investigators and analysts are increasingly experiencing large, even terabyte sized data sets when conducting digital investigations. State-of-the-art digital investigation tools and processes are efficiency constrained from both system and human perspectives, due to their continued reliance on overly simplistic data reduction and mining algorithms. The extension of data mining research to the digital forensic science discipline will have some or all of the following benefits: (i) reduced system and human processing time associated with data analysis; (ii) improved information quality associated with data analysis; and (iii) reduced monetary costs associated with digital investigations. This paper introduces data mining and reviews the limited extant literature pertaining to the application of data mining to digital investigations and forensics. Finally, it provides suggestions for applying data mining research to digital forensics.

Keywords: Digital forensics, digital investigation, large data sets, data mining

1. Introduction

The digital forensic discipline is experiencing heightened importance and attention at a time when data storage requirements are increasing exponentially. Enterprise storage needs are predicted to increase sevenfold between 2003 and 2006, and email volume is increasing at an annual rate of 29% [12]. Hinshaw [18] reports that data rates are doubling every nine months – twice as fast as Moore's Law. Because of this exponential growth, it is not uncommon for larger corporations and law enforcement agencies to face digital investigations with data sets as large, or larger than a terabyte [24, 28].

Current digital investigation tools cannot handle terabyte-sized data sets in an efficient manner [25]. Their overall efficiency is constrained

by the employment of simple hashing and indexing algorithms. Current digital forensics tools and processes are simply not scalable to large data sets [7, 14, 26]. Even with moderately large data sets (i.e., 200 gigabytes), data extraction and analytic activities become inordinately slow and inefficient. Processing times for limited keyword searches (10-20 keywords) can take days, and the human analyst is overwhelmed with the number of "hits" to review.

Digital investigations are also hindered by the limited processing capabilities of human analysts. As data sets increase in size, the amount of data required for examination and analysis also increases. This obviates the digital investigator's ability to meticulously review all keyword search "hits," files by file type, or all applicable system logs. It is therefore imperative that the digital investigation process be improved.

Currently, digital investigation processes and tools underutilize computer processing power through continued reliance on simplistic data reduction and mining algorithms. In the past, when human labor was cheap and computers were expensive, the analytical burden was shifted to analysts. For quite some time, the roles have been reversed, yet the digital forensics field has continued to levy the preponderance of its analytical burden on the human analyst.

Digital forensics is not the only discipline faced with the task of sifting through massive volumes of data. Other disciplines have employed data mining techniques to resolve this problem. However, little research has focused on applying these techniques to criminal forensics, and even less on digital forensics. The purpose of this paper is to increase awareness of data mining techniques within the digital forensic community and to show how they can be utilized to solve large data set challenges.

The following section provides a brief tutorial of data mining. Next, Section 3 surveys data mining research as applied to digital forensics. Section 4 suggests ways in which data mining techniques can be applied to digital forensics. This is followed by a discussion in Section 5 of the benefits and limitations of extending data mining to digital investigations. Concluding remarks are provided in Section 6.

2. Data Mining

Data mining embodies a multi-disciplinary approach to finding and retrieving information, and relies on several reference disciplines that enjoy long, rich research streams, including mathematics, statistics, computer science, and information science. Techniques developed and/or used within these disciplines (e.g., artificial intelligence, machine learning, pattern recognition, data visualization, and database processes) are

utilized to develop data models, identify patterns, detect anomalies, and retrieve information.

Data mining processes, methods and techniques can be divided into three major classes: descriptive modeling, predictive modeling and content retrieval. Descriptive modeling summarizes or discriminates data, whereas predictive modeling identifies characteristics that can help predict future observations. Both modeling techniques largely necessitate structured data (i.e., databases, XML documents and graphs). Content retrieval data mining is perhaps the most complex type of data mining, particularly because it extracts information from complex and/or semi-structured/unstructured data sets. Content retrieval techniques are typically directed toward text data, multimedia data (e.g., image, video and audio data), World Wide Web data, spatial data, time-series or sequential data and complex objects (containing more than one data type). Han and Kamber [15] and Hand, *et al.* [16] provide excellent discussions regarding each of these classes.

2.1 Descriptive Data Modeling

Descriptive data mining relies on data generalization and conceptualization to generate descriptions that facilitate both characterization (summarization) and comparison (discrimination). Characterization techniques tend to overlap with data warehouse techniques, which cleanse, transform and summarize large data sets into smaller data sets with aggregate level information. Since aggregation inherently results in data loss, characterization data mining techniques will likely have limited utility in digital forensic investigations, but may prove very helpful in non-forensic digital investigations, such as internal corporate investigations and military operations.

Comparison techniques, also known as discrimination, produce a set of rules for comparing the general features of objects found within two or more data collections. As a generic example, one may want to determine the general characteristics of customers who frequently make on-line purchases, compared with those who rarely make on-line purchases. The techniques used for data discrimination are very similar to those used for data characterization, except they include some form of comparative measure. The aggregation function associated with characterization techniques is not necessarily applicable to comparison techniques, thus comparison techniques may have greater potential in digital forensic investigations than characterization techniques.

2.2 Predictive Data Modeling

Predictive data mining builds on descriptive data mining, in that the first step is inherently descriptive. However, the ultimate goal is to anticipate and/or categorize future observations (data). There are three primary sub-classes of predictive data modeling: association rule-based data analysis, regression and classification. Association rule-based data mining attempts to identify relationships or "associations" between items or item features. Given a set of items, association analysis establishes rules that can predict the occurrence of an item, based on the occurrence of other items in the transaction. Thus, they capture the probability that two data items co-occur, facilitating the profiling of co-occurring items with high frequencies and probabilities. Similarly, anomalies can be detected by finding co-occurring items within the data set that have a very low probability of co-occurring.

Regression-based data mining is used when data observations (response or dependent variable) can be modeled, and therefore predicted, by a mathematical function using a given set of data characteristics (predictor or independent variables). The mathematical function may be linear or non-linear. For example, weight (Y) can be modeled as a linear function of height $(X) : Y = \alpha + \beta X$.

Classification techniques represent the third and final sub-class of predictive data mining techniques. While introduced last, these techniques are potentially the most relevant to the present discussion. Classification techniques are used for both descriptive and predictive data mining – the primary difference being the purpose for which they are employed. Classification data mining techniques employ a whole host of methods to classify data, including: decision tree induction, Bayesian classification/belief networks, neural networks, nearest neighbor classification, genetic algorithms, case-based reasoning, rough sets and fuzzy logic. Each of these methods offers different ways to develop classification schemes to describe data and subsequently classify future observations (data). Classification rules, decision trees and/or mathematical formulae may be generated to facilitate prediction.

Cluster analysis (or clustering) is a frequently used classification technique, and thus bears specific mention. All classification techniques employ a training (or learning) phase and a validation phase. During the training phase, data is analyzed to support the development of the classification scheme. The learning may be "supervised" or "unsupervised." In supervised learning, the class labels are predefined in the data set, and thus the classes as well as the number of classes are predefined. In unsupervised learning, nothing is predetermined – the classification

technique itself establishes classes and number of classes based on similarities and dissimilarities found in the data. Cluster analysis employs unsupervised learning via partitioning methods, hierarchical methods, density based methods, grid-based methods, and model-based methods. Cluster analysis is often used for anomaly and outlier detection, and is applicable to intrusion detection and fraud detection.

2.3 Content Retrieval Data Mining

The third major class of data mining techniques is content retrieval. Whereas descriptive and predictive data mining techniques largely leverage mathematical and statistical reference disciplines, content retrieval depends heavily on research in information science and computer science, particularly in the areas of information retrieval, artificial intelligence, machine learning and natural language processing. Content retrieval methodologies are geared toward retrieving content from unstructured or semi-structured data sets (e.g., a text document or set of images, as opposed to a structured database). The primary sub-classes of content retrieval are: information (text) retrieval, multimedia data mining, web mining, complex data object mining, spatial data mining and time-series/sequential data mining.

Text retrieval is often referred to as information retrieval, simply because information retrieval goals and objectives have historically been text related – it is only recently (relatively speaking) that the desire to mine other types of content has emerged. The goals of information retrieval are usually to compare documents, rank importance or relevance of documents, or find patterns/trends across multiple documents. Common information retrieval techniques fit into two major categories: keyword-based (similarity-based using terms) and indexing-based. Index-based approaches such as latent semantic indexing are more prevalent, due to current limitations of natural language processing algorithms,[1] and the inherent ability of indexing approaches to be more conceptually based.

Multimedia data mining techniques are particularly relevant to digital forensics in the realm of image retrieval. Images can be analyzed and retrieved using description-based retrieval systems that use keywords, captions, size, creation time, etc., or using content-based retrieval systems that use color histograms, wavelet transformations, or measures of texture, shape, or objects. In content-based retrieval systems, image representations are created using a variety of methods, including: (i) feature representation via abstract pixel data; (ii) 3-D color feature vectors spatially averaged over the entire image; (iii) k-dimensional color his-

tograms with subsequent partitioning using clustering algorithms; (iv) 3-D texture vectors using coarseness/scale, directionality and contrast; and (v) 20-dimensional shape feature vectors using area, circularity, eccentricity and axis orientation.

Using such techniques, images can be classified as containing humans, buildings, etc. and retrieved accordingly. Other content retrieval techniques are directed at web data, complex data objects, spatial data and time-series/sequential data. Each has potential application to digital forensics, such as the applicability of time-series/sequential data mining techniques to network log data in network forensics cases and the applicability of web mining to retrieve content, structure and usage data from World Wide Web (WWW) based data.

3. Data Mining and Digital Investigations

A basic understanding of data mining illuminates its potential application to digital investigations. Data mining techniques are specifically designed for large data sets – attempting to find and retrieve data and otherwise hidden information amongst voluminous amounts of data. The data may or may not be structured, noisy or from the same source. In digital forensics, data sources are both structured and unstructured; noisy and not noisy; and from both homogeneous and heterogeneous sources-particularly in large data set cases.

Large data set research, specifically research related to data mining, has yet to be extended to digital forensics to any appreciable degree. We argue that the lost potential associated with this continued research void is analogous to the Internet without *Google* (data indexing and querying), the credit card industry without fraud detection (data mining) algorithms, and Wal-Mart without its 500 terabyte data warehouse to facilitate customer relationship management (CRM) and retailer-supplier decision support. As we collectively strive to strengthen the science of digital forensic and investigations [22], it is imperative that researchers begin to leverage and extend large data set research – particularly in the area of data mining.

Large data set research has only been extended to digital forensics in a handful of instances. The following subsections describe the various data mining techniques employed to date.

3.1 Predictive Data Modeling

3.1.1 Classification via Cluster Analysis. de Vel, *et al.* [13] utilized a Support Vector Machine learning algorithm to mine e-mail content and positively identify its authorship from a set of exemplars

from known authors. This procedure is intuitively akin to handwriting analysis, although its accomplishment requires much more complex processing. A Support Vector Machine learning algorithm is a classification-based data mining algorithm that seeks to categorize data based on certain key features of the data. In this instance, categories refer to different authors. Distinguishing features of the e-mail content and headers are used to classify each e-mail in the proper category according to its author.

3.1.2 Classification via Discriminant Analysis. Carney and Rogers [6] demonstrated how stepwise discriminant analysis can be used to determine the probability of intentionality associated with downloading contraband images (i.e., child pornography). Their motivation was to provide a mechanism for event reconstruction with calculable accuracy probability to help investigators investigate the "Trojan defense."[2] They examined seven different characteristics (variables or features) of the data and empirically determined that a single model using two features (average difference between file creation times and median difference between file creation times) can be developed and used to ascertain user intentionality associated with the incidence of contraband stored on digital media.

3.1.3 Association Rule Mining. de Vel collaborated with Abraham [1] and Kling [2] to profile user behavior and identify behavioral irregularities using system activity logs. These researchers applied association rule data mining to network and system data to determine association rules based on system interaction history of the user. They developed activity-based and event-based association rules that related user role (e.g., system administrator vs. financial analyst) to typical system activities (e.g., scan the internal network vs. review company financial statements). In doing so, they were able to develop behavioral profiles of typical users, and thereby mine subsequent log data sets for anomalies (e.g., someone using the account of a financial analyst to scan the internal network).

3.1.4 Content Retrieval Incorporating Text Mining. Text mining (also referred to as "information retrieval") is an outgrowth of the information science discipline that has enjoyed several decades of research advances in computational linguistics, which facilitate text categorization, semantic extraction and content summarization [30]. Text mining has received expanded research attention in recent years due to significant increases in business intelligence demands and data avail-

ability [30]. Shannon [27] developed a text mining technique called the Forensic Relative Strength Scoring (FRSS), which is basically a data categorization algorithm applicable to data reduction and extraction activities. The FRSS uses a scoring system with two measures: ASCII proportionality and entropy score (ASCII data tends to have less entropy and non-ASCII data). Using FRSS, Shannon [27] demonstrated how these two measurers could be used for data reduction and extraction of ASCII data. Shannon recommended further research be conducted using N-gram-based text categorization techniques [8] to subsequently identify the language and category (email, news posting, technical document, etc.) of the extracted ASCII data.

3.1.5 Peripheral Data Mining Research. Other data mining research has been applied peripherally to digital forensics and digital investigations. A large research stream has developed regarding the application of data mining techniques to intrusion detection, with particular emphasis on anomaly detection versus signature-based intrusion detection. A few recent examples include research by Barbara, *et al.* [3], Mukkamala and Sung [21], and Stolfo, *et al.* [29].

Data mining techniques have also been extended to image analysis (counterfeit detection [23] and steganography detection [19]), as well as data visualization to facilitate link analysis. Finally, data mining techniques have been extended to crime data analysis to profile criminals and identify criminal networks [9–11, 17, 31, 32]. These extensions of data mining theory demonstrate the vast potential data mining has for the digital forensics and digital investigation disciplines.

4. Data Mining in Digital Investigations

We are urging greater investigation into the use of data mining techniques to aid digital investigations in two primary areas: crime detection and crime investigation. Descriptive, predictive and content retrieval data mining techniques can be loosely mapped to each of these areas. We purport that such mapping and application will result in: (i) reduced system and human processing time associated with data analysis; (ii) improved analytical effectiveness and information quality; and (iii) reduced monetary costs associated with digital investigations.

4.1 Crime Detection

Crime detection activities involve behavioral profiling and anomaly detection (both behavioral and technological). Descriptive modeling and predictive modeling techniques are thus applicable, while content

retrieval data mining techniques are not. Descriptive modeling incorporating characterization (summarization) techniques are applicable, for example, in determining conformability of a data set to Benford's Law [20], thereby supporting economic fraud detection upon analyzing electronic data. Characterization techniques could also be used to better focus limited investigative resources. Such techniques, for example, could show which computers were used more for certain types of activity, such as electronic communication.

Descriptive modeling incorporating comparison (discrimination) techniques can be used to determine the similarity of two non-identical objects (with non-matching hashes), such as images in steganography detection, or source code in intellectual property theft detection. Comparison techniques are also applicable when comparing user (account) data from network logs. Stark dissimilarity where none is expected might be indicative of unauthorized activity.

While the characterization and comparison descriptive modeling techniques described above are applicable to crime detection data mining, predictive modeling techniques, e.g., association-based rule mining and classification data mining, are more commonly applied. Association-based rule mining and classification data mining are inherently designed to describe similarities amongst data observations and occurrences. As a result, dissimilarities (or anomalies) clearly emerge. Anomaly detection based on association rule mining has obvious applications in the areas of network intrusion detection, fraud detection, and unauthorized use detection (i.e., in espionage cases), and have already been introduced.

4.2 Crime Investigation

Crime investigation activities map to the Data Analysis Phase of the digital investigations process. The Data Analysis Phase consists of three sub-phases: data surveying, data extraction, and data examination [4]. Data mining techniques can assist with all three sub-phases, and the applicability of data mining to the Data Analysis Phase may often result in a natural progression through the data mining classes (descriptive $\rightarrow$ predictive $\rightarrow$ content retrieval).

During the data survey sub-phase, descriptive (characterization) modeling techniques can be employed to profile use and activity (e.g., percent free space, percent free space wiped, percent ASCII vs. binary and percent data by file type). During the data extraction sub-phase, classification data mining techniques can be used to reduce the amount of data for analysis. For example, before information retrieval techniques are employed, a data set can be reduced to ASCII data. It is important to

emphasize that data reduction techniques should be classification-based as described, as opposed to descriptive characterization (summarization) techniques, due to data loss associated with the latter.

The applicability of data mining techniques to the data extraction sub-phase continues and overlaps with the data examination sub-phase. Key crime investigation activities include: entity extraction [11], content retrieval [15, 16, 30], and crime data analysis (or link analysis) [5, 17, 32]. Entity extraction refers to predictive modeling based classification techniques that are geared toward identification. Identification may be a function of person, user account, email account, authorship, personal characteristics, etc. This technique can be a useful mechanism for investigators seeking attribution.

Content retrieval has clear and extensive applicability to digital investigations, such as mining large data sets for text documents containing specific content or involving particular individuals, or mining large data sets for contraband graphic images (e.g., child pornography, counterfeit currency). Taking a closer look at the former example, the goal of text (information) retrieval is usually to compare documents, rank importance or relevance of documents, or find patterns/trends across multiple documents. Each of these goals is extensible to digital investigations – particularly the latter two. Ranking the importance or relevance of documents relative to investigative objectives, criminal allegations, or target content facilitates data extraction during the Data Analysis Phase and minimizes, as well as prioritizes, the "hits" an investigator or analyst has to review. This is critical when dealing with large data sets. Finding patterns and trends across multiple documents assists an investigator in profiling users and uncovering evidence for which exact keywords are unknown.

Content retrieval data mining is also of use in the areas of multimedia mining, web mining and time/spatial data mining. Multimedia mining is particularly useful from the standpoint of image detection involving contraband, counterfeit and steganographic images. Web mining classifies and retrieves content, structure and usage data from World Wide Web (WWW) based data. This technique could be very useful for digital forensic investigators. In addition to being voluminous, web data is exceptionally "noisy" relative to the investigative objectives at hand. The sheer volume and "noisiness" of this data is absolutely overwhelming and incompatible with manual data analysis techniques. Finally, time/spatial data mining may have applicability in chronology analyses in network and media investigations.

The last key crime investigation activity – crime data (link) analysis – demonstrates the data mining class progression that often occurs

during the Data Analysis Phase. The goal of crime data analysis is to identify and visualize associations amongst social (criminal) networks. Three primary steps are associated with crime data analysis: transformation, sub-group detection, and association/pattern visualization. During data transformation, data from disparate sources is converted via characterization based descriptive data mining techniques to develop a basic "concept space" for the data. The results are then fed into the sub-group detection step, wherein cluster analysis (predictive) data mining techniques are used to identify groups of criminal actors, e.g., those that communicate with one another. Associations between sub-groups are then identified, including network form (e.g., star shaped) and node (criminal actor) centrality/criticality. Predictive data mining techniques (social network analysis approaches, such as block-modeling) are employed to accomplish this [11]. Content retrieval techniques may or may not play a role in crime data (link) analysis, but the progressive application of multiple data mining classes and techniques is evident.

5. Discussion

Employing data mining techniques to aid digital forensic investigations has many potential advantages. If applied properly it meets the three-fold goal of (i) reducing system and human processing time; (ii) improving the effectiveness and quality of the data analysis; and (iii) reducing cost. Other benefits include better utilization of the available computing power, as well as improved ability to discover patterns/trends normally hidden to the human analyst. There are, however, potential limitations of employing data mining techniques to aid in digital forensic investigations. First and foremost, the techniques are virtually untested within the digital forensic discipline. Since data mining has enjoyed previous success, however, there is reason to believe it will be effective in this discipline. Second, there is a lack of understanding of data mining techniques by the digital forensics and digital investigations community.

Other limitations include: (i) evaluation of content retrieval algorithms is inherently subjective, thus different analysts may view success differently; (ii) data mining inherently converts data to a much higher level of abstraction, therefore uncertainty and error calculations are particularly relevant; and (iii) the application of data mining techniques may require advanced knowledge and training of analysts and investigators. We argue that these limitations do not limit the potential of extending data mining research to digital forensics and digital investigations. Instead, these limitations merely reflect the challenges associated with doing so.

6. Conclusions

Clearly, data mining techniques can support digital investigations in myriad ways. However, much work needs to be done before these techniques can be successfully applied and disseminated throughout the digital investigation community. Suggestions for promoting these techniques include: increasing the awareness and understanding of data mining techniques, training digital investigators in the use of these techniques, and creating a framework for using these techniques in digital investigations.

Originally, it was our intention to create such a framework, but it soon became clear that research and awareness of data mining techniques, as applied to digital investigations, is in its infancy. We therefore encourage other researchers and practitioners to assist us in improving awareness and skills in this area. Terabyte-sized data sets are already challenging analysts and investigators. Therefore, an active stream of research extending data mining research to digital forensics and digital investigations is desperately needed.

Notes

1. Natural language processing (NLP) techniques have trouble overcoming polysemy (multiple meanings for the same term) and synonmy (multiple terms with the same meaning).

2. The "Trojan defense" is a criminal defense that argues the defendant did not intentionally engage in the illegal activity, but rather that a Trojan, virus, or hacker was responsible for the illegal activity.

References

[1] T. Abraham and O. de Vel, Investigative profiling with computer forensic log data and association rules, *Proceedings of the IEEE International Conference on Data Mining*, pp. 11-18, 2002.

[2] T. Abraham, R. Kling and O. de Vel, Investigative profile analysis with computer forensic log data using attribute generalization, *Proceedings of the Fifteenth Australian Joint Conference on Artificial Intelligence*, 2002.

[3] D. Barbara, J. Couto, S. Jajodia and N. Wu, ADAM: A testbed for exploring the use of data mining in intrusion detection, *ACM SIGMOD Record*, vol 30(4), pp. 15-24, 2001.

[4] N. Beebe and J. Clark, A hierarchical objectives-based framework for the digital investigations process, to appear in *Digital Investigation*, 2005.

[5] D. Brown and S. Hagen, Data association methods with applications to law enforcement, *Decision Support Systems* vol. 34, p. 10, 2002.

[6] M. Carney and M. Rogers, The Trojan made me do it: A first step in statistical based computer forensics event reconstruction, *Digital Evidence*, vol. 2(4), p. 11, 2004.

[7] E. Casey, Network traffic as a source of evidence: Tool strengths, weaknesses and future needs, *Digital Investigation*, vol. 1, pp. 28-43, 2004.

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

[9] M. Chau, J. Xu and H. Chen, Extracting meaningful entities from police narrative reports, *Proceedings of the National Conference for Digital Government Research*, pp. 271-275, 2002.

[10] H. Chen, W. Chung, Y. Qin, M. Chau, J. Xu, G. Wang, R. Zheng and H. Atabakhsh, Crime data mining: An overview and case studies, *Proceedings of the National Conference for Digital Government Research*, p. 4, 2003.

[11] H. Chen, W. Chung, J. Xu, G. Wang, Y. Qin and M. Chau, Crime data mining: A general framework and some examples, *IEEE Computer*, vol. 37(4), pp. 50-56, 2004.

[12] Connected Corporation, Storage reduction facts and figures (www.connected.com/downloads/Items_for_Downloads/Storage Facts_Figures.pdf).

[13] O. de Vel, A. Anderson, M Corney and G. Mohay, Mining e-mail content for author identification forensics, *ACM SIGMOD Record*, vol. 30(4), pp. 55-64, 2001.

[14] J. Giordano and C. Maciag, Cyber forensics: A military operations perspective, *Digital Evidence*, vol 1(2), pp. 1-13, 2002.

[15] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Academic Press, San Diego, California, p. 550, 2001.

[16] D. Hand, H. Mannila and P. Smyth, *Principles of Data Mining*, MIT Press, Cambridge, Massachusetts, 2001.

[17] R. Hauck, H. Atabakhsh, P. Ongvasith, H. Gupta and H. Chen, Using Coplink to analyze criminal justice data, *IEEE Computer*, vol. 35, pp. 30-37, March 2002.

[18] F. Hinshaw, Data warehouse appliances: Driving the business intelligence revolution, *DM Review Magazine*, September, 2004.

[19] J. Jackson, G. Gunsch, R. Claypoole and G. Lamont, Blind steganography detection using a computational immune system: A work in progress, *Digital Evidence*, vol. 1(4), pp. 1-19, 2003.

[20] G. Moore and C. Benjamin, Using Benford's Law for fraud detection, *Internal Auditing*, vol. 19(1), pp. 4-9, 2004.

[21] S. Mukkamala and A. Sung, Identifying significant features for network forensic analysis using artificial intelligence techniques, *Digital Evidence*, vol. 1(4), pp. 1-17, 2003.

[22] G. Palmer, A Road Map for Digital Forensics Research: Report from the First Digital Forensics Research Workshop, Technical Report DTR-T001-01 Final, Air Force Research Laboratory, Rome, New York, 2001.

[23] F. Petitcolas, R. Anderson and M. Kuhn, Information hiding: A survey, *Proceedings of the IEEE*, vol. 87(7), pp. 1062-1078, 1999.

[24] D. Radcliff, Inside the DoD's Crime Lab, *NetworkWorldFusion*, pp. 1-5, March 8, 2004.

[25] V. Roussev and G. Richard III, Breaking the performance wall: The cases for distributed digital forensics, *Proceedings of the Digital Forensics Research Workshop*, pp. 1-16, 2004.

[26] M. Schwartz, Cybercops need better tools, *Computerworld*, p. 1, July 31, 2000.

[27] M. Shannon, Forensics relative strength scoring: ASCII and entropy scoring, *Digital Evidence*, vol. 2(4), pp. 1-19, 2004.

[28] P. Sommer, The challenges of large computer evidence cases, *Digital Investigation*, vol. 1, pp. 16-17, 2004.

[29] S. Stolfo, W. Lee, P. Chan, W. Fan and E. Eskin, Data mining based intrusion detectors: An overview of the Columbia IDS Project, *ACM SIGMOD Record*, vol. 30(4), pp. 5-14, 2001.

[30] D. Sullivan, *Document Warehousing and Text Mining: Techniques for Improving Business Operations, Marketing and Sales*, John Wiley, New York, p. 542, 2001.

[31] G. Wang, H. Chen and H. Atabakhsh, Automatically detecting deceptive criminal identities, *Communications of the ACM*, vol. 47(3), pp. 71-76, 2004.

[32] J. Xu and H. Chen, Fighting organized crimes: Using shortest-path algorithms to identify associations in criminal networks, *Decision Support Systems*, vol. 38, pp. 473-487, 2004.

Chapter 2

FORENSICS AND PRIVACY-ENHANCING TECHNOLOGIES

Logging and Collecting Evidence in Flocks

Martin Olivier

Abstract Flocks is a privacy-enhancing technology (PET) used to hide the web usage patterns of employees in an organization against profiling or mere inspection by administrators and other officials. However, Flocks is intended to support the identification of senders of malicious requests by means of a legitimate forensic investigation.

This paper formalizes what should be logged for an appropriate forensic investigation. Also, it considers exactly what evidence should be explored once a malicious request has been noticed. It argues that (i) evidence that would have been collected about a malicious request if the PET were not used, should still be collected, and (ii) evidence that becomes visible by some legitimate means because the PET is used, should be collected. However, information that has not become visible by such legitimate means, but is available because the PET is being used, should not be collected. In the latter case, privacy concerns override the fact that a malicious request might be uncovered by investigating more logged information. These positions are defended and formalized using mathematical notation.

Keywords: Privacy-enhancing technologies, logging, evidence collection

1. Introduction

The relationship between Privacy-Enhancing Technologies (PETs) and forensics has always been an uncomfortable one. The former is used to keep information about individuals private. The latter is used to uncover information about crimes or other relevant incidents. Often the information relevant to these two areas overlap: It is possible to argue that an individual who commits a crime gives up his or her right to pri-

vacy – at least as far as data about the crime is concerned. On the other hand, even if a demonstrable crime has been committed, the privacy of individuals who have not been demonstrated to be involved with the crime, should not be compromised. Two examples of this uneasy relationship include the infamous Clipper chip [19] and the `anon.penet.fi` case [8] that led to improvement of remailers to the point where they made forensic investigations extremely difficult [11].

We previously proposed a PET – referred to as Flocks – that allows anonymous web browsing in an organizational context [16]. It uses a number of proxies that randomly forward requests to one another, or – with some probability α – to the destination server outside the organization.[1] The intention was to propose a technology that provides an acceptable level of anonymous browsing, but one that allows a request to be associated with a sender under appropriate conditions. The previous paper [16] focused on the two primary parameters that influence the operation of Flocks should be chosen. The two parameters are α, the probability with which a request will be forwarded to the (real) destination server, and N, the number of proxies in the system. One of the explicit considerations to determine these parameters was the need to be able to conduct a forensic investigation. However, the exact information logged at each proxy for such an investigation was not considered.

The current paper has a dual purpose. First, it addresses the question of exactly what should be logged in Flocks to enable a forensic investigation. This question is answered by expressing the log contents using mathematical notation. Clearly, such a formal specification will contribute to the accuracy of analysis when the need for analysis arises. Second, the paper uses this formalized log content and explores which parts of it should be used (exposed) during a specific forensic investigation. From the discussion, the paper proposes a position that intends to allow legitimate forensic investigations, but preserves privacy to the greatest possible compatible degree once an investigation is underway.

The remainder of the paper is organized as follows. Section 2 describes the required background about PETs in general, and Flocks in particular. Section 3 considers logging in Flocks and formalizes the content of logs. Section 4 considers the information that should be uncovered during a forensic investigation and (formally) proposes the position that the paper supports. Section 5 presents the conclusions.

2. Background

Over the years a number of PETs have been introduced. Many of the current ideas were already present in Chaum's 1981 notion of a mix

[7]. In the mid 1990s attention turned to specific technologies used on the Internet (and, more specifically), the Web. This focus was based on the realization that interacting on the Internet often leaves a trail that may be used to learn more about an individual than should be tolerated. The PETs developed in the 1990s were mostly intended to allow the individual to exert control over what information is made known to other parties, by using appropriate intermediaries. These intermediaries could be (fixed) third parties, such as Anonymizer [6], Janus (or Rewebber) [22] or LPWA [10]. The third parties could also be (randomly or deterministically) selected from a set of available proxies or routers. Such ideas were used in Crowds [21] and Onion routing [12].

In more recent times, attention has turned to PETs that can be employed inside an organization to help the organization protect the information it has collected about individuals. Examples of developments in this regard include Hippocratic databases [1] and E-P3P [2, 14]. It has been argued that the costs (to the organization) associated with deploying such a PET will be fully recovered by customer satisfaction – and even lead to increased business opportunities [13, 20, 24]. In the case of Flocks a similar case could be made about the benefits employee satisfaction holds for the organization.

Flocks [16] was introduced as a PET based on technologies such as Crowds, but one intended for deployment within an organization. In this environment it was intended to minimize external traffic by caching web pages retrieved from the Internet as far as possible, but yet minimized an administrator's (or even a manager's) ability to breach the privacy of users by just browsing logs (or more actively profiling users). However, a fundamental tenet behind Flocks was the fact that the PET does not reduce users' accountability – and that forensic investigations should be possible where a legitimate reason exists for such an investigation. In essence, Flocks operates as follows. Each user operates a personal proxy and acts as trustee for the logs generated by that proxy. When a user submits a request, it is submitted to this proxy. The proxy serves the request from cache, if possible. Else it forwards the request to the external destination, with probability α. If it does not forward the request to the destination, it forwards it to another proxy that is chosen randomly. In the latter two cases, the result is cached (if possible), once it arrives. Requests from other proxies are dealt with similarly.

Anonymous remailers predate many of the other PETs mentioned here. Anonymous remailers have highlighted some of the most fundamental issues of collection of evidence weighed against the user's right to privacy [8, 11].

A system such as Flocks where logging is explicitly enabled assumes that these logs will be stored in a manner where they will not be abused. Such a PET will be useful when profiling of users and browsing of logs are the main threats. Where true anonymity is desired, a stronger form of PET will be required. The legal aspects of obtaining information from logs such as those maintained by Flocks have been discussed by others [5, 8, 9] and fall outside the scope of the current paper.

Many overviews of PETs have been published [9, 11, 17, 23]. For a structured view of PETs, see the Layered Privacy Architecture [15].

The computer forensic process is often divided into preparation, collection, analysis and presentation phases. For an overview of the forensic process, see the series of articles on the topic by Wolfe [25–30].

3. Logging in Flocks

The information to be logged by each proxy in a Flocks system has been implied in our earlier paper [16]. However, to conduct a proper forensic investigation in such a system, it is necessary to consider exactly what is logged. This section therefore formalizes this.

Consider some node $n_i \in P$ where P is the set of Flocks proxies in a given environment. Assume some message $m = \langle s, d, r \rangle$ arrives at some time t from some source s, where d is the (eventual) destination of the message and r is the request for a web page. The standard action for web proxies is to log $\langle t, s, d, r \rangle$. We will accept this standard behaviour as our initial solution.

Next, consider how this message is "rewritten" when it is forwarded. The request and destination remain unchanged. Expressed as it would be in Z-like notation, this could be written formally as $r' = r$ and $d' = d$. The proxy now acts as the source of the message; hence $s' = i$. The time of the new message is determined by the receiving system and (at best) we know $t' \approx t$. Note that clocks on proxies and servers will often (usually) differ significantly.

According to the way Flocks is designed, node n_i can forward the request (with probability α) to destination d, or to some node n_j. We argue that little will be gained by logging which choice has been made and, in the latter case, which node n_j has been selected. This will be discussed below when we discuss forensic investigations.

We represent the log at each node n_i as L_i where L_i is a relation over $T \times S \times D \times R$, where T is the set of possible times at which a request can occur (including date), $S = P$ is the set of nodes that could have (directly) sent or forwarded a message to the current proxy. Let D be the set of destinations; for simplicity we assume that $D \cap P = \emptyset$. R is the

set of possible requests; essentially R is the set of valid strings that form HTTP requests. We assume that the current practice of logging the first line of the full HTTP request is used – and hence that R is the set of such (potential) first lines. However, this can be expanded to include more of the request without affecting the remainder of the paper.

Below it will be useful to also consider an "augmented" log L_i^+ that consists of the quadruples of L_i, with the value i (the identity of the particular proxy whose log is being considered) added as a fifth column. Clearly, the domain of this column is the set of proxies P. However, for ease of reference we introduce another set $I = P$ as the domain of this fifth dimension. Therefore L_i^+ is a set of tuples from $T \times S \times D \times R \times I$.

Let us briefly consider the reliability of logged information. Clearly, s can be spoofed. However, given the fact that Flocks is cast in an environment where HTTP is used, the transport layer protocol used will be TCP. Therefore, a three-way handshake will be performed for any connection attempt [18]. What this implies is that host s has to have responded to the second part of the handshake before the request will be logged. This does not mean that the address of s cannot be spoofed, but that it will be somewhat harder to do this; in particular, if some node n_q masquerades as node s, n_q will have to be able to respond to appropriate messages to s; to some extent n_q will have to have taken control of the address of s. Therefore the log entry pointing to s will be somewhat more reliable than if UDP were used, but should still not be fully trusted. To address this problem it might be worth considering signed requests. Signed requests will n ot only help to determine the authenticity of the source of the request, but will also address non-repudiation: If an entry occurs in a log that states that the query has been received from another proxy, but no trace is found of the query in that proxy's log, it will be hard to determine which log is correct. Although signed requests will solve this problem, signing is not necessary to address the problems considered by this paper. We therefore leave signing of log entries for future research.

It has also been noted that the time logged depends on the particular server or proxy. This can be addressed to some extent by normalizing the times: If a "heartbeat" request is sent to each proxy at fixed intervals, and those heartbeats logged, such entries may be used to adjust to logged time to correspond in all logs to be used for an investigation. We do not consider this option further in this paper; in fact the paper will argue that time is not as important in this environment as it seems initially.

Given the logged information as described in this section, we now turn our attention to using this information for a forensic investigation.

```
10.229.33.5 - - [12/Mar/2002:14:00:01 +0200] "GET /default.ida?NNN
   NNNN
```

messages can indeed be intercepted. Many countries have laws that allow – or even require – the interception of messages under specific conditions. We assume that (social and technical) mechanisms exist that prevent recording of all messages, because such wide scale recording of messages will clearly render the PET useless. In particular do we assume that interception of messages are governed by an appropriate l aw and organizational policies that serve to protect the individual user.

As an example of this second case assume that an organization has a policy that employees are not allowed to visit pornographic websites. Monitoring the DNS lookup log files is a relatively unobtrusive way to verify that this policy is complied with. Suppose it is found that a DNS lookup occurred for `www.xx.co.za`. Suppose this site is known to contain pornographic material. The node that performed the DNS lookup will be the last proxy in the chain of proxies used. Examining that proxy's log file will indicate whether a full investigation is warranted, and the examination can proceed from there.

In both cases (internal and external) it is possible to precisely prescribe the log entries that are required for the investigation. In fact, without such a characterization of the entries required for the investigation, an investigation should be frowned upon – and the different custodians of the logs should make such searches – at least – very difficult to perform.

We assume that the log entries in an appropriate investigation can be characterized by a (possibly compound) condition c. In the case of an attack (such as the Code Red example used above), the request may be enough to characterise the log entries; here c will be of the form $R = \rho$ for some specific value ρ. In other cases a specific request sent to a specific host will be the cause of an investigation; hence, c will be of the form $(R = \rho)\&(D = \delta)$ for specific values of ρ and δ. More options exist for reasonable compositions of c; such conditions will, however, not be discussed in this paper.

Note that we will use relational algebra below to manipulate the log information. The R and D used in condition c above are used there to indicate the column that should be used when testing whether the condition is satisfied. In this context T, S, D, R and I will be used to identify the columns that have the (like-named) sets T, S, D, R and I as their domains.

In all cases it should be possible to identify the proxy from which the suspicious request has been sent. If the suspicious entry appears in an external log file, the last proxy that forwarded it will be identified as the sender. If the message is intercepted on the internal network, the sender at the point of interception will be one of the proxies. However,

we assume that this proxy is not part of condition c, but is identified separately as node x.

4.2 Investigation by Log Amalgamation

The first option to consider to investigate the incident is to amalgamate all the log files and search in the compound log file for suspicious log entries. Let Λ be this compound log file, with

$$\Lambda = \cup_{i \in I} L_i^+$$

The suspicious entries can then be isolated by selecting those for which condition c holds (with σ as the relational *select* operator):

$$\sigma_{c \& (I=S)}(\Lambda)$$

The additional condition $(I = S)$ ensures that requests are selected at their origin: Since we assume that all users run a proxy and send their own requests to their own proxies, all requests from node i will be logged with i as source in log L_i.[3]

There are, however, two privacy-related reasons why this solution is not acceptable.[4]

First, amalgamating all the log files is against the spirit of using multiple proxies in the first place: Once the logs have been amalgamated, much that does not concern the incident being investigated can be learnt from the log files. This may partly be addressed by performing an appropriate select prior to amalgamating the log files. This, however, still leads to the second reason why this solution is not acceptable.

This second reason is related to reasonable grounds to invade a person's communication privacy. The fact that evidence has been found that implicates one user does not imply that the browsing habits of all users who have submitted a similar request can necessarily be searched – and the query given above will find all such requests – and hence identify all originating senders. This does not mean that such a search is never warranted; however, we contend that it should not be the default mode of investigation. This is supported by RFC 3227 [4]: "Do not intrude on people's privacy without strong justification."

4.3 Investigating by Following the Chain

A more suitable approach will be one that starts at the implicated proxy and follows the chain back to the originating node. However, even this simple algorithm can be interpreted in different ways, and therefore needs to be formalized. Consider the following: Suppose log L_x indicates that it received the the suspicious request from some node y. If log L_y

indicates that the request was received from node z, one simply has to follow the link to L_z. However, it is possible that more than one "matching" request may exist in L_y. Suppose, for example, that the suspicious entry in L_x contains $e_1 = \langle t, y, d, r \rangle$, with y as the source (and t, d and r the time, destination and request — as used elsewhere in the paper). Now suppose L_y contains $e_2 = \langle t_2, z, d, r \rangle$. Note that d and r are identical in both entries. Does e_2 match e_1? Does the answer depend on the relationship between t and t_2? Clearly, if $t = t_2$ there is a case to be made that the two entries match. However, since different logs are based on different clocks, it is unlikely that the two times will be exactly the same. (It is even possible – albeit improbable – that an entry $e_3 = \langle t, z, d, r \rangle$ does not match e_1 exactly.) One solution will be to use $t \approx t_2$ as criterion, with some appropriate decision about how close two time values should be to be approximately equal.

Let us consider when two times are approximately close enough. If a page was requested by some node y from node x it usually means that y would not have requested the page immediately before this. (If it had, it would normally have cached it and would not have requested it again.) Similarly, if y requested a page at time t there will be no need to request it again soon after t (because y will still have it cached then). Hence, times do not have to be as accurate as in other [3] environments: t and t_2 may be considered approximately equal even if they differ by tens of minutes or more. (Obviously there are exceptions: Some pages cannot be cached and in some cases a page will be requested just prior to the expiration of the cache copy and just after that. However, these cases seem few enough to work with the general case until empirical evidence proves that a different approach is required.) [5]

4.4 Our Position

- Malicious requests that would probably have been noticed if the PET were not used, but were hidden by the PET, should be exposed for examination. The malicious requests exposed, should be limited to those that benefitted directly or indirectly from the noticed request.

- Malicious requests that are noticed because the PET is being used, should cause other requests that, in principle, could have caused the noticed malicious request to be issued, to be exposed. However, other requests (that is other from those that could have lead to the issuing of the malicious request) that benefitted from the fact that the malicious request has indeed been issued, should not be exposed.

It seems that the possible ambiguity in this description is unavoidable (unless the description is extended significantly). We therefore formulate it in mathematical terms.

POSITION 1 (UPSTREAM TRACKING) *Suppose that request ρ to destination δ from proxy x has been flagged. Let E be the set of log entries whose examination is justified. Then*

$$\sigma_{(D=\delta)\&(R=\rho)}(L_x^+) \subseteq E \tag{1}$$

and, if $\langle t, s, d, r, i \rangle \in E$ then

$$\sigma_{(D=\delta)\&(R=\rho)}(L_s^+) \subseteq E \tag{2}$$

Equation 1 ensures that the entries in the log of the implicated proxy (x) that could have forwarded request ρ to server δ are included. Equation 2 extends this recursively to include all the other entries earlier in the chain that, eventually, led to this final query.

This part of the specification will ensure that all the implicated entries are in E. The following restriction will ensure that no unnecessary entries are in E.

POSITION 2 (USE LIMITATION OF UPSTREAM ENTRIES) *Suppose that entry $e_1 = \langle t_1, s_1, d_1, r_1, i_1 \rangle \in E$. Then, if e_1 is not the entry that was directly implicated, there must exist an $e_2 = \langle t_2, s_2, d_2, r_2, i_2 \rangle \in E$ such that $d_1 = d_2 = \delta$, $r_1 = r_2 = \rho$ and $s_2 = i_1$. This clearly implies the existence of a chain of entries $e_1, e_2, \ldots, e_k$ with $k \geq 1$. The chain is terminated when the flagged entry is reached, i.e., k has to exist with $e_k = \langle i_k, t_k, s_k, d_k, r_k \rangle \in E$, $i_k = s$, with s the source from which the flagged request was sent, $d_k = \delta$ and $r_k = \rho$.*

Where Position 1 states that log entries should be followed towards their origin, Position 2 states that only log entries that could have lead to the request that caused the investigation, should be included.

The nodes that should be investigated are those from which the request originated, i.e., from $\Pi_I(\sigma_{I=S}(E))$, where Π is the *project* operator.

COROLLARY 1 (PROHIBITION OF DOWNSTREAM TRACKING) *As should be clear from the examples, Positions 1 and 2 imply that trails could be followed from the point of (valid) interception to its (possible multiple) "origins," but not towards its destination. By implication it cannot be followed from any point between the point of interception and the destination to other possible "origins." The formal requirements given above,*

already make provision for this. However, to stress the point consider two chains (as used above) $e_1, e_2, \ldots, e_k, \ldots, e_m$ *and* $e'_1, e'_2, \ldots, e'_n$ *with* $e'_n = e_m$*. Assume* e_k *is implicated, with* $1 \leq k < m$*. We argued that, although the evidence in* e_1 *and* e'_1 *are linked (via* $e'_n = e_m$*), this does not offer justification to use* e'_1*. Clearly, if other grounds exist,* e'_1 *can be used; such grounds will exist if there exists an* $e'_j = \langle i'_j, t'_j, s'_j, d'_j, r'_j \rangle \in E$ *such that* $i'_j = x$*,* $d'_j = \delta$ *and* $r'_i = \rho$ *and the communication has been observed legitimately at* x *– that is, it can be used if the evidence points directly to* e'_j *and* e'_j *points to* e'_1 *as its origin request.*

Arguably the last requirement is the most contentious of our position. However, we do not explore the implications that not accepting it will have on the investigation process in the current paper.

4.5 Time

Times at which requests have been issued have played a somewhat paradoxical role in our discussion thus far. On the one hand, it was assumed during the initial discussion that times can indeed be correlated – an assumption that was known to be unrealistic. However, in Positions 1 and 2 time played no explicit role. This was based on a number of arguments, the most important of which was the premise that the fact that a PET is being used, should not decrease accountability.

If the case can be made that time is not significant in cases where it is easy to correlate log entries, it clearly is possible to make a similar case where time is (more realistically) hard to correlate. Hence we accept that time is not one of the significant factors that will be used to correlate logs when conducting a forensic examination in Flocks.

However, time cannot be totally ignored: Should a malicious request that has just been reported, uncover a similar request that was issued a year ago? It is possible that the request issued a year ago was not malicious at the time, but changed circumstances make it appear malicious in the current environment. In addition to this possible objection, processing logs that stretch over years can be very expensive – especially when they can affect the results by introducing noise.

We therefore suggest that time plays a role when the logs for investigation are extracted. L_x should not normally be the log that was compiled over the entire lifespan of x, but some practical subset – one that is larger than apparently required for the investigation. We suggest that the time period to be taken into account should be determined at the start of the investigation and only changed if reasons for change are found. Empirical evidence is required to determine the effect of using generous log files for the investigation.

4.6 Importance of the Destination

In the example and positions above, the destination of the request was seen as an important facet of the investigation. It is, however, possible that a reported incident should not necessarily be associated with a given destination. A virus attack from some computer using the PET and that is noticed by some server δ does not mean that only those attacks that were launched on δ should be investigated; it will be prudent to investigate all attacks. The discussion should therefore be read as based on some (justifiable) condition c, that could include destination, request and other aspects – alone or in combination.

Therefore, if the destination is not a significant part of the investigation, the parts of the conditions using δ may be omitted in all the equations used in Section 4.

5. Conclusions

This paper considers issues that should be taken into account when a forensic investigation is conducted in the Flocks environment. It motivates what should be logged, and describes positions that govern the manner in which log file data is amalgamated for the purposes of the investigation. While these positions might be transferable to other PET environments, the argument depends on the caching that is done in Flocks. Therefore, care should be taken when generalizing these positions to other PETs.

Future work includes an empirical study on the effects of using logs for longer or shorter periods and the use of signed log entries to address spoofing and repudiation. It also remains to be formalized how the data collected should be analyzed to tie the evidence to a particular user, as well as appropriate presentation of this evidence.

Acknowledgements

This work is supported by the National Research Foundation under Grant 2054024, as well as by Telkom and IST through THRIP. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author and therefore the NRF, Telkom and IST do not accept any liability thereto.

Notes

1. The analogy with Crowds [21] should be clear.
2. Note that the format of the server log entry differs in a number of respects from the proxy log entries discussed in the previous section; amongst others, the order of the fields are different.

3. If each node does not run its own proxy, originating hosts can be identified as those for which $s \notin I$.

4. This "solution" presents a third problem, but one that is easily solved by not allowing proxies to forward requests to themselves: In the original Flocks proposal, the possibility of a proxy forwarding a request to itself has not been excluded. If we use the suggested method to identify the origin of a message, forwarding by a proxy to itself cannot be allowed. Since such forwarding does not seem to be useful, we assume that it will not occur, and explicitly require from a Flocks proxy not to forward a request to itself.

5. For an example that illustrates some of the subtleties of matching entries, please see the extended version of this article, which is available at `http://mo.co.za/abstract/flfor.htm`.

References

[1] R. Agrawal, J. Kiernan, R. Srikant and Y. Xu, Hippocratic databases, *Proceedings of the Twenty-Eighth International Conference on Very Large Databases*, 2002.

[2] P. Ashley, S. Hada, G. Karjoth and M. Schunter, E-P3P privacy policies and privacy authorization, *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pp. 103-109, 2003.

[3] C. Boyd and P. Forster, Time and date issues in forensic computing – A case study, *Digital Investigation*, vol. 1(1), pp. 8-23, 2004.

[4] D. Brezinski and T. Killalea, Guidelines for evidence collection and archiving, RFC 3227, The Internet Society, February 2002.

[5] I. Brown and B. Laurie, Security against compelled disclosure, *Proceedings of the Sixteenth Annual Computer Security Applications Conference*, pp. 2-10, 2000.

[6] M. Caloyannides, Encryption wars: Shifting tactics, *IEEE Spectrum*, vol. 37(5), pp. 46-51, 2000.

[7] D. Chaum, Untraceable electronic mail, return addresses and digital pseudonyms, *Communications of the ACM*, vol. 24(2), pp. 84-88, 1981.

[8] G. Du Pont, The time has come for limited liability operators of true anonymity remailers in cyberspace: An examination of the possibilities and the perils, *Journal of Technology Law & Policy*, vol. 6(2), pp. 175-217, 2001.

[9] A. Froomkin, Flood control on the information ocean: Living with anonymity, digital cash and distributed databases, *University of Pittsburgh Journal of Law and Commerce*, vol. 395(15), 1996.

[10] E. Gabber, P. Gibbons, D. Kristol, Y. Matias and A. Mayer, Consistent, yet anonymous, web access with LPWA, *Communications of the ACM*, vol. 42(2), pp. 42-47, 1999.

[11] I. Goldberg, D. Wagner and E. Brewer, Privacy-enhancing technologies for the Internet, *Proceedings of the Forty-Second IEEE International Computer Conference*, pp. 103-109, 1997.

[12] D. Goldschlag, M. Reed and P. Syverson, Onion routing, *Communications of the ACM*, vol. 42(2), pp. 39-41, 1999.

[13] IBM, Privacy in a connected world (www-1.ibm.com/industries/government/doc/content/bin/private.pdf), 2002.

[14] G. Karjoth, M. Schunter and M. Waidner, Platform for Enterprise Privacy Practices: Privacy-enabled management of customer data, *Proceedings of the Second International Workshop on Privacy Enhancing Technologies*, 2003.

[15] M. Olivier, A layered architecture for privacy-enhancing technologies, *South African Computer Journal*, vol. 31, pp. 53-61, 2003.

[16] M. Olivier, Flocks: Distributed proxies for browsing privacy, in *Proceedings of SAICSIT 2004 – Fulfilling the Promise of ICT*, G. Marsden, P. Kotzé and A. Adesina-Ojo (Eds.), pp. 79-88, 2004.

[17] Organization for Economic Cooperation and Development (OECD), Inventory of privacy-enhancing technologies (PETs), Report DSTI/ICCP/REG(2001)1/FINAL, 2002.

[18] J. Postel, Transmission control protocol, RFC 793, Defense Advanced Research Projects Agency, Fairfax, Virginia, 1981.

[19] D. Price, Micro View – Clipper: Soon a de facto standard? *IEEE Micro*, vol. 14(4), pp. 80-79, 1994.

[20] PrivacyRight, Control of personal information: The economic benefits of adopting an enterprise-wide permissions management platform (www.privacyright.com/info/economic.html), 2001.

[21] M. Reiter and A. Rubin, Anonymous web transactions with Crowds, *Communications of the ACM*, vol. 42(2), pp. 32-48, 1999.

[22] A. Rieke and T. Demuth, JANUS: Server anonymity in the worldwide web, *Proceedings of the EICAR International Conference*, pp. 195-208, 2001.

[23] V. Seničar, B. Jerman-Blažič and T. Klobučar, Privacy-enhancing technologies: Approaches and development, *Computer Standards & Interfaces*, vol. 25, pp. 147-158, 2003.

[24] Wave Systems, User managed privacy: A new approach for addressing digital privacy and personal information on the Internet (www.wave.com/technology/PrivacyWhitePaper.pdf), 2000.

[25] H. Wolfe, Evidence acquisition, *Computers & Security*, vol. 22(3), pp. 193-195, 2003.

[26] H. Wolfe, Evidence analysis, *Computers & Security*, vol. 22(4), pp. 289-291, 2003.

[27] H. Wolfe, Encountering encryption, *Computers & Security*, vol. 22(5), pp. 388-391, 2003.

[28] H. Wolfe, Presenting the evidence report, *Computers & Security*, vol. 22(6), pp. 479-481, 2003.

[29] H. Wolfe, Forensic evidence testimony – Some thoughts, *Computers & Security*, vol. 22(7), pp. 577-579, 2003.

[30] H. Wolfe, Setting up an electronic evidence forensics laboratory, *Computers & Security*, vol. 22(8), pp. 670-672, 2003.

Chapter 3

A NETWORK-BASED ARCHITECTURE FOR STORING DIGITAL EVIDENCE

Mark Davis, Gavin Manes and Sujeet Shenoi

Abstract The storage and handling of digital evidence are creating significant challenges for federal, state and local law enforcement agencies. The problems include acquiring and processing massive amounts of digital evidence, maintaining the integrity of the evidence, and storing digital evidence for extended periods of time. This paper describes a network-based storage architecture that helps address these issues. The architecture also supports collaborative efforts by examiners and investigators located at geographically dispersed sites.

Keywords: Storage media, network area storage, storage area networks

1. Introduction

Law enforcement agencies are facing major challenges with regard to the storage and processing of digital evidence [6, 17]. Complex cases are being encountered that require evidence to be extracted from networks, multi-drive computers and sophisticated portable electronic devices [14, 15]. Most cases still involve single hard drives, but hard drive capacities can be very large [1, 11]. In a recent case, the Tulsa (Oklahoma) Police Department's Cyber Crimes Unit seized a personal computer with three 250GB hard drives. New hard drives were purchased to handle the large volume of data. However, the unit's imaging workstations relied on ATA-100 technology, which could not support drives larger than 137GB. New equipment based on ATA-133 technology had to be purchased so that the larger hard drives could be used to process evidence.

The long-term storage of digital evidence is also presenting serious problems for law enforcement agencies [6, 14, 17]. Sometimes, evidence has to be maintained only for the duration of a trial. In other instances, evidence must be stored for the length of the sentence. A recent triple

homicide case in Tulsa involved more than 350GB of digital evidence. The 27 year-old accused received a life sentence without parole, which could require that all the evidence in the case be stored for 50 years or more. Digital storage media degrade over time and few, if any, media can guarantee the integrity of the stored evidence beyond fifteen years [4, 16, 17]. Special environmentally-controlled storage rooms can help extend the life of certain media, but these are very expensive.

Meanwhile, digital media technology is constantly changing. Currently, it is difficult to obtain a 5.25" floppy drive, although it was the primary removable storage medium just fifteen years ago. Evidence stored on an IDE hard drive may not be accessible twenty years from now because the hardware might not be readily available [17].

Evidence handling – especially maintaining the chain of custody – is a strict and meticulous process that requires special consideration with regard to digital evidence [10]. Digital evidence is easily moved and copied, making it difficult to document who had access to the evidence and when the evidence was accessed. Moreover, digital evidence must be protected using physical access controls as well as computer-based access controls [2]. Since most law enforcement agents are not computer security experts, it can be difficult for them to ensure that the integrity of the evidence is maintained.

Digital forensic procedures must also be reliable enough to withstand courtroom scrutiny. Law enforcement agents compute hash values of image files to verify their integrity, but problems arise when the integrity of an image is lost. In such cases, the original storage media must be re-imaged [10, 17]. However, the media may not always be available or it may be damaged or destroyed.

The sheer volume of evidence involved in many cases requires examiners and investigators, who may be at different geographic locations, to cooperate in digital forensic investigations. What is needed is an efficient methodology for storing, moving and examining data across geographic boundaries. The ideal implementation is a centralized repository where evidence is stored and maintained, but which allows the evidence to be securely accessed from remote locations. Furthermore, the system must be technologically transparent and it should eliminate the need for forensic examiners and investigators to perform systems administration duties.

This paper describes a network-based solution for storing and handling large quantities of digital evidence. The design is intended to streamline digital forensic investigations and support the collaborative analysis of digital evidence at multiple locations. To provide a framework for discussing the network-based storage solution, the following

section describes the main technologies for implementing networks with massive storage capabilities.

2. Digital Evidence Storage Networks

Two main technologies exist for implementing networks with massive storage capabilities: network area storage (NAS) and storage area networks (SAN). These technologies are discussed below.

2.1 Network Area Storage

Network area storage (NAS) is a solution for storing massive quantities of data in a centralized location [13]. NAS grew out of the file server concept made popular by Netware and Microsoft's Windows NT server [5]. The realization that comprehensive operating systems were not needed to perform storage functions led to the creation of NAS storage devices. These storage devices, with embedded operating systems, are attached to a network and accessed via standard protocols, e.g., TCP/IP. Access control is typically implemented by a network sharing mechanism similar to Windows shares or Samba shares in UNIX [9].

Due to its ease of use, NAS became a popular digital evidence storage solution. In the late 1990s, some FBI laboratories relied on NAS-based SNAP appliances – small rack mountable devices with proprietary operating systems that contain 250GB to 15TB of storage [8]. However, as the protocols for accessing and analyzing digital evidence became more complicated, a more scalable solution than NAS was deemed necessary.

2.2 Storage Area Networks

A storage area network (SAN) is a segmented area of a network that handles storage and data transfer between computers and storage elements [3, 12, 13]. The SAN model removes storage devices and storage-heavy traffic from general networks, creating a network designed exclusively for storage operations. SANs use fibre channel or fabric networks to implement many-to-many connectivity between servers and storage devices. The network-based architecture of SANs makes them highly configurable and scalable, and able to support redundancy.

The addressing scheme used in a fabric network requires that every network device have a unique world wide name (WWN). A WWN is a 64-bit hexadecimal number coded into each device, similar to a MAC address on an Ethernet network. A logical unit number (LUN) is a name given to a RAID set within a storage array. A software client allows LUNs within the disk array to be assigned to WWNs on the network, enabling a LUN to behave identically to a local hard drive on

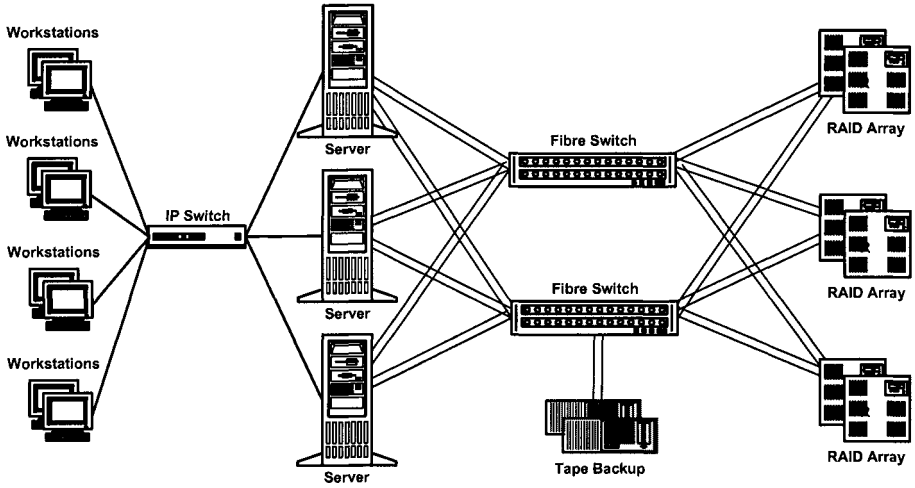


Figure 1. Storage area network.

a computer. LUNs can be re-assigned, unassigned or even increased in size dynamically according to network needs.

Figure 1 shows a typical SAN architecture. RAID disk arrays in the SAN are attached to one or more fibre switches, which in turn are connected to fibre channel cards within computers in the network. Connecting RAID arrays and computers to more than one fibre switch ensures that the SAN and disk arrays have redundant paths to access data during hardware failures. RAID disk arrays speed up data transfer and provide data integrity and redundancy in the event of an accidental loss of digital evidence.

2.3 Combining NAS and SAN Technology

NAS and SAN are similar technologies and either can work well in a given situation [5, 9, 13]. Both technologies use RAID arrays to store data. In a NAS implementation, almost any machine in a LAN can connect to a NAS storage device. However, in a SAN implementation, only computers equipped with fibre channel cards may connect directly to storage devices. A NAS device handles operating system data, backup and mirroring, and data tranfers using operating system metadata (e.g., file name and byte information). On the other hand, a SAN addresses and transfers data by raw disk blocks. A NAS allows data sharing between multiple operating systems (e.g., Unix and Windows NT); a SAN only allows sharing via fibre channel and access is dependent on operating system support. Finally, a NAS typically manages its own file system, while SAN file systems are managed by connected servers.

By combining SAN and NAS technology, LUNs can be shared by user workstations via servers (see Figure 1). This approach is often used by web and file server applications for which availability and load balancing are primary concerns. Usually a portion of the SAN is assigned to a server, which provides services to many clients [3, 12]. The server need not have local storage, resulting in significant cost savings. Server applications, operating systems and storage are easily reassigned, shared or moved from one server to another. For example, if a server fails, its LUN can be reassigned to another server. A LUN can even be assigned to several servers, which means only one copy of the data exists and all the servers would be identical. This also allows many servers and workstations to access and process data simultaneously.

Implementing such a system in a digital forensic environment can drastically improve operational efficiency. Forensic examiners do not have to keep hundreds or thousands of hard drives in evidence storage lockers to preserve evidence. Furthermore, data is transported quickly and easily by reassigning LUNs to different servers [8]. A SAN eliminates the need to manually transport evidence – data is simply assigned wherever it is needed.

The efficiency of a NAS over SAN solution is verified by statistics from the FBI's North Texas Regional Computer Forensics Laboratory (NT-RCFL) [7, 8, 18]. During the four and a half months following the September 11, 2001 attacks, NT-RCFL processed approximately 7.4TB of data using fifteen dedicated examiners. After the NT-RCFL's SAN became operational a year later, an 8.5TB case was processed in one month using only five dedicated examiners. The SAN also helped reduce case backlogs. With its original NAS-based SNAP solution, NT-RCFL had accumulated eight months of case backlog as of September 2001. The NT-RCFL SAN increased data examination rates by a factor of five – the number of examiners fell from fifteen to twelve and the case backlog dropped to just two months.

The NAS over SAN model is an ideal evidence storage solution for a large FBI laboratory, which typically processes and maintains digital evidence at a single location. On the other hand, many federal, state and local law enforcement agencies employ smaller facilities at multiple locations. This requires digital evidence to be delivered, examined and processed at one location, and then physically transported to another location for further examination, presentation or storage. To streamline digital forensic investigations, it is necessary to design a modified NAS over SAN model that facilitates the collaborative analysis of digital evidence at geographically dispersed sites.

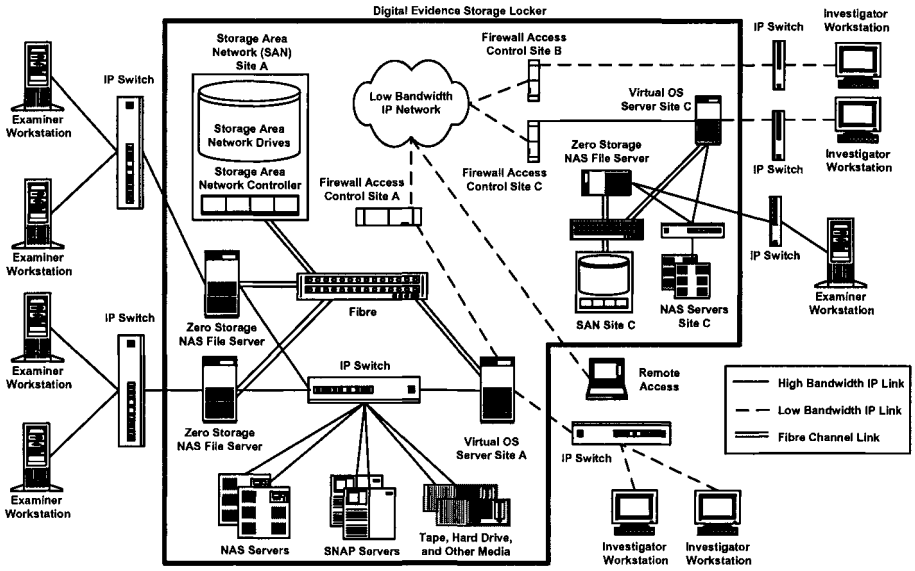


Figure 2. Digital evidence custodian architecture.

3. Digital Evidence Custodian

This section describes the architecture of a digital evidence custodian (DEC), which is intended to streamline investigations and support the collaborative analysis of digital evidence at multiple locations. In many computer crime investigations, the same individual serves as the examiner, investigator and case agent. However, this situation is rapidly changing as caseloads and evidence volume increase, and digital forensic tasks become highly specialized. To streamline investigations, the DEC architecture implements the logical and physical separation of duties of forensic examiners, forensic investigators and evidence custodians.

Figure 2 presents the DEC architecture. Evidence is stored in a digital evidence storage locker (DESL), which primarily uses NAS over SAN technology to facilitate the collaborative processing of digital evidence by examiners, investigators and case agents who may be at different locations. Any storage technology, e.g., NAS, SNAP servers, tape drives or file servers, may be used in a DESL. The DEC architecture eliminates network configuration, administration and maintenance tasks and provides transparency of technology to forensic specialists and agents, enabling them to focus exclusively on case investigations.

Forensic examiner workstations in the DEC architecture (Figure 2) are dedicated computers for imaging storage media. These computers are networked to a zero storage local server in a NAS configuration

to access storage devices located within the internal SAN configuration. The internal NAS and SAN systems comprise the storage locker (DESL).

During an investigation, the DEC dedicates storage space for a case within the DESL to an examiner's zero storage local server. The zero storage local servers share this space with the examiners' workstations, allowing them to image storage media. Depending on the urgency of the case, a forensic examiner may perform analysis functions such as live previews, file filtering and keyword searches on the imaged media. Upon completion of the imaging process, the evidence is stored in the DESL and all access to the evidence is removed from the examiner. This reduces, if not eliminates, the tedious drive-swapping imaging process that is common in digital forensics practice.

Full examination of the evidence is accomplished by assigning the desired section of the DESL to a virtual OS server. The virtual OS server provides access to evidence stored within the DESL and the primary platform for evidence processing. The DEC creates a session on a virtual OS server, assigns permissions to evidence in the DSL, and configures the desired forensic programs and examination environment. The virtual OS server assigns this access in a write-protected mode, allowing traditional examination of digital evidence using forensic software. Alternatively, the virtual OS server may place evidence in a persistent mode, allowing examiners to view and handle evidence as if it were in the original imaged device. Once the examination is complete, access to the evidence is removed from the virtual OS server; this secures the evidence within the DESL, which models a physical evidence custodian and evidence locker.

At DESL locations, access to the virtual OS server is accomplished via secure firewalls and VPN connections over TCP/IP networks (Figure 2). The standard IP network infrastructure provides examiners and investigators from other locations with access to digital evidence and examination reports via broadband or even low-bandwidth modem connections. Digital evidence can be mirrored to other DESL sites to support data redundancy and parallel examinations.

Figure 3 shows a DEC designed to support electronic crimes investigations and digital evidence storage needs of the Oklahoma State Bureau of Investigation (OSBI). OSBI has three main sites (Tulsa, Oklahoma City and Weatherford), each of which could house a full-blown DESL, including a SAN, virtual OS server and digital forensic workstations. Each site would field two to five digital forensic examiners who would serve the entire state of Oklahoma. The three DESLs would be connected by dedicated high-speed Internet2 connections, allowing agents from the three main sites to collaborate on cases. For example, if Weatherford has a small caseload, agents in Weatherford could work on Tulsa cases

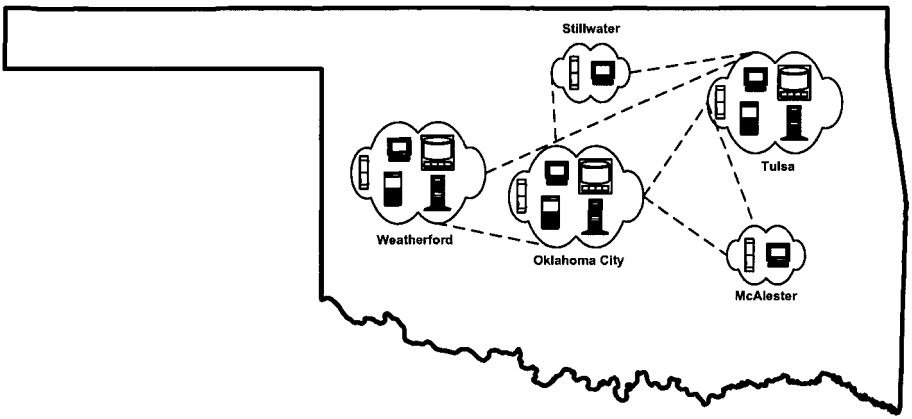


Figure 3. Digital evidence custodian supporting state-wide access.

without making the three-hour trip to Tulsa. Evidence in high priority cases could be processed at all three locations simultaneously. Furthermore, digital evidence could be mirrored at multiple sites to enhance efficiency and support redundancy and disaster recovery efforts.

OSBI agents at other locations in the state could also participate in digital forensic investigations. For example, agents in Stillwater and McAlester (Figure 3) could access the network of DESLs using smaller networks of forensic workstations and servers. Agents at these locations could perform imaging, examinations and report generation, significantly enhancing the overall productivity.

The OSBI DEC could support investigations throughout Oklahoma. For example, law enforcement agents from a rural community with limited expertise and technology could seize storage media, computers or portable electronic devices and send them to a DESL site or to a location with access to a DESL for processing. They could then access the results and investigative reports using standard Internet connectivity.

4. Conclusions

The digital evidence custodian (DEC) architecture is a powerful, yet relatively inexpensive, network-based solution for storing and handling massive quantities of digital evidence. In a typical implementation, evidence is stored in a network of digital evidence storage lockers (DESLs), which use NAS over SAN technology and dedicated high-speed Internet2 connections to facilitate the collaborative processing of digital evidence by examiners, investigators and case agents who may be at different locations. Use of standard IP network infrastructures enables other authorized individuals to access digital evidence and examination reports

maintained in the DESL network via broadband or even low-bandwidth modem connections. In addition to simplifying the tasks of storing evidence and maintaining its integrity, the DEC architecture significantly enhances the productivity of digital forensic investigations by supporting the distributed access and processing of digital evidence.

References

[1] M. Anderson, Hard disk drives – Bigger is not better, New Technologies (www.forensics-intl.com/art14.html), 2001.

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

[3] T. Clark, *Designing Storage Area Networks: A Practical Reference for Implementing Fibre Channel and IP SANs (Second Edition)*, Addison-Wesley, Reading, Massachusetts 2003.

[4] Digital Preservation Coalition, Media and formats (www.dpconline.org/graphics/medfor/media.html).

[5] B. Goldworm, The difference between SANs and NAS, *Network World Storage in the Enterprise Newsletter* (www.networkworld.com/newsletters/stor/2000/1002stor1.html?nf), 2000.

[6] Institute for Security Technology Studies, *Law Enforcement Tools and Technologies for Investigating Cyber Attacks: Gap Analysis Report*, Dartmouth College, Hanover, New Hampshire, 2004.

[7] T. Maiorana, Building community support - The Heart of America RCFL, presented at the *Digital Forensics Working Group Meeting*, Orlando, Florida, March 2004.

[8] C. Mallery, Personal correspondence, North Texas Regional Computer Forensics Laboratory, Dallas, Texas, February 20, 2004.

[9] NAS-SAN.com, Technology overview (www.nas-san.com/differ.html), Zerowait Corporation, Newark, Delaware, 2003.

[10] B. Nelson, A. Phillips, F. Enfinger and C. Steuart, *Computer Forensics and Investigations*, Thompson Course Technology, Boston, Massachusetts, 2004.

[11] D. Orzech, Rapidly falling storage costs mean bigger databases, *CIO Update: Technology Trends* (www.cioupdate.com/trends/article.php/2217351), June 4, 2003.

[12] C. Poelker and P. Nikitin, *Storage Area Networks for Dummies*, Wiley, New York, 2003.

[13] W. Preston, *Using SANs and NAS*, O'Reilly, Sebastopol, California, 2002.

[14] RCFL National Program Office, *Regional Computer Forensics Laboratory Program: Fiscal Year 2004 Annual Report*, Federal Bureau of Investigation, Quantico, Virginia, 2005.

[15] M. Reith, C. Carr and G. Gunsch, An examination of digital forensic models, *International Journal of Digital Evidence*, vol. 1(3), 2002.

[16] K. Shanmugasundaram, A. Savant, H. Bronnimann and N. Memon, ForNet: A distributed forensics network, *Proceedings of the Second International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security, Lecture Notes in Computer Science, Volume 2776*, Springer-Verlag, Berlin-Heidelberg, Germany, pp. 1-16, 2003.

[17] M. Vatis, *Law Enforcement Tools and Technologies for Investigating Cyber Attacks: A National Need Assessment*, Institute for Security Technology Studies, Dartmouth College, Hanover, New Hampshire, 2002.

[18] R. Voss, Building a team – Chicago RCFL, presented at the *Digital Forensics Working Group Meeting*, Orlando, Florida, March 2004.

Chapter 4

DIGITAL FORENSICS: MEETING THE CHALLENGES OF SCIENTIFIC EVIDENCE

Matthew Meyers and Marcus Rogers

Abstract This paper explores three admissibility considerations for scientific evidence currently engaged in U.S. courts: reliability, peer review and acceptance within the relevant community. Any tool used in a computer forensic investigation may be compared against these considerations, and if found wanting, evidence derived using the tool may be restricted. The ability to demonstrate the reliability and validity of computer forensic tools based on scientific theory is an important requirement for digital evidence to be admissible. A trusted third party certification model is discussed as an approach for addressing this issue.

Keywords: Computer forensic tools, evidence, admissibility

1. Introduction

It should come as no surprise that there is a dramatic increase in digital evidence being brought before courts in the United States and elsewhere in the world. As a result, courts are becoming concerned about the admissibility and probative value of digital evidence. Meanwhile, the discipline of computer forensics appears to be struggling over methods and practices that will meet the courts' "standards" for scientific evidence. For the purpose of this discussion, the term "computer forensics" is defined as the use of an expert to preserve, analyze and produce data from volatile and non-volatile storage media.

The admissibility of evidence in U.S. federal courts and many state courts is based on the Federal Rules of Evidence (F.R.E.) [17] and upon various Supreme Court opinions that interpret its constitutional and legal application. F.R.E. Rule 702 specifically addresses the testi-

Table 1. Daubert and F.R.E. 702 criteria.

Daubert	F.R.E. 702
(1) such testimony was admissible only if relevant and reliable	(1) can be and has been tested
(2) the Federal Rules of Evidence (FRE) assigned to the trial judge the task of insuring that an expert's testimony rested on a reliable foundation and was relevant to the task at hand	(2) has been subjected to peer review or publication
(3) some or all of certain specific factors—such as testing, peer review, error rates, and acceptability in the relevant scientific community—might possibly prove helpful in determining the reliability of a particular scientific theory or technique	(3) has (a) high known or potential rate of error, relevant to the scientific community – where such factors are reasonable measures of the testimony's reliability; the trial judge may ask questions of this sort not only where an expert relies on the application of scientific principles, but also where an expert relies on skill or experience-based observation

mony of experts concerning scientific evidence and is applicable to computer forensics. Several opinions guide the application of F.R.E. 702, among them are the Daubert [18] and Kumho Tire [19] decisions. In the Daubert decision, the court specifically held that Frye [13] was superseded by F.R.E. and several judicial considerations were identified. This paper focuses on the application of Daubert to F.R.E. 702, and its potential impact on the field of computer forensics.

The Daubert decision defines the role of the judge as a gatekeeper tasked with filtering out "junk science." However, in practice, this filtering usually involves attorneys raising Daubert challenges – contesting the qualifications of an expert, the scientific nature of their evidence, and the validity and reliability of the methods and hardware/software tools (e.g., write blockers and software suites) employed. If a tool is successfully challenged, derivative evidence from the tool may not be admissible or, at the very least, is given less weight in deliberations by the judge and/or jury.

The Daubert ruling recognizes that judges, when determining the scientific validity of the method or reasoning in question, are faced with many considerations. This paper examines the considerations of reliability, peer review and acceptance as outlined in Daubert, as well as the applicable sections of F.R.E. (see Table 1) to determine if computer forensics tools meet the consideration for acceptance as scientific evidence. The paper concludes by proposing a solution that meets at least some of the considerations.

2. Reliability and Validity

To demonstrate reliability and validity under Daubert, a number of factors must be taken into consideration: known or potential error rates, testing, and commonly agreed upon methods. Unfortunately, computer forensic tools and techniques fall short of meeting these considerations. Currently, there is a strong reliance by practitioners on proprietary software whose error rates are unknown. Vendors, protective of their market share, have not published information concerning error rates or even the exact reasons for minor and major version changes. Furthermore, the forensic community may be prevented from conducting in depth tests by terms imposed by software licenses and legislation such as the Digital Millennium Copyright Act [16]. While there is some limited testing for error rates and reliability of certain products by third parties, e.g., by the U.S. National Institute of Standards and Technology (NIST) [7], these bodies do not assume liability for the results and do not certify or accredit specific tools. Published results pertaining to these tests take several months to become available and are usually based on technologies, tools or applications that have been superseded by newer releases.

Given the lack of information and the restrictions on full error testing and reporting, indirect approaches are used to demonstrate the validity, integrity and reliability of digital evidence. For example, an investigator typically computes a digital signature or hash value for the original evidence (e.g., media, partition, drive) and for the bit-stream image of the original source; the two values can be compared at any time to demonstrate that they match. The algorithms used to compute signatures and hash values provide mathematical assurances that if the values match, the image has not been corrupted or contaminated, and has a high degree of fidelity relative to the original source (a true copy), and can be considered as best evidence [8, 14]. While this approach can determine if an error occurred, it provides no information about the error source or about the actual or potential error rates.

3. Peer Review

One of the Daubert considerations is whether an expert's methods, processes and underlying reasoning have been peer reviewed and/or published. The rationale behind this consideration is that if the implementation of a theory is flawed, the results will be flawed and, by peer review or publication, others in the scientific community will have the opportunity to discover flaws and supply recommendations or resolve errors prior to the implementation or acceptance of the theory. The corollary is that tools used to derive the results should also be peer reviewed. Computer

forensic tools automate the basic manual processes of evidence acquisition, examination and analysis, and in some cases, reporting. Indeed, in computer forensics there tends to be a heavy reliance on tools and, as some have suggested, on blind faith. This has lead to an industry myth that certain tools have been accepted by the courts. However, the courts have ruled that an inanimate object (e.g., a software package) cannot be considered an expert [20]. This does not necessarily imply that the tool or the results obtained using the tool cannot be included in scientific testimony. What it does mean is that the individual who used the tool may have to testify about the procedures used and about the reliability of the tool prior to the results being admitted as evidence.

It has been suggested that the use of open source tools satisfies the peer review consideration and may increase the reliability of digital evidence derived from the use of these tools [4]. Proponents of the open source movement have stated that because end users can examine (peer review) the source code, it is more secure and, therefore, more reliable. However, the mere ability to view the source code does not translate to better security or to meeting the requirements of reliability, testing and peer review [9]. Furthermore, open source code is often the work of several authors who may or may not be trustworthy and who may or may not follow state-of-the-art software engineering methodologies. Also, the code can be altered at any time, including after formal testing for error rates. Thus, the courts may find that open source tools do not meet the scientific considerations. Simply put, open source does not in and of itself mean that it is peer reviewed: Who are the peers? Where was the source code published (e.g., journals, conferences)? The potential for the source code to be reviewed does not equate to it actually being peer reviewed.

The exact nature of peer reviewing and vetting by way of publication is problematic in general, and not just for open source tools. Few publications directly address computer forensic methods and processes. At the time of writing this paper, there were only two quasi peer-reviewed journals dedicated to computer forensics: *International Journal of Digital Evidence* and *Journal of Digital Investigation.* In reviewing the Daubert considerations, it is unclear whether peer review requires publication in journals or presentation at conferences focusing on the particular field in question. Given the precedent set by other forensic sciences, e.g., forensic psychology and DNA analysis, the lack of such journals and conferences at the very least does not support the inference of peer vetting, and the reliability and validity of scientific methods and reasoning.

4. General Acceptance

Yet another important consideration mentioned by the U.S. Supreme Court in Daubert is whether a tool, technique or principle has "attracted widespread acceptance within a relevant scientific community." This presumes two elements: (i) there is a relevant scientific community, and (ii) the community has a generally accepted set of principles or processes. Since computer forensics is a relatively new field, it may not have an established scientific community *per se*. While the American Society of Crime Laboratory Directors – Laboratory Accreditation Board (ASCLD-LAB) has recognized computer forensics as a scientific subdiscipline, other professional bodies such as the American Academy of Forensic Sciences (AAFS) have not formally done so. To date U.S. courts have not commented on this fact. However, with defense attorneys becoming technically sophisticated, it is possible that the recognition of the field and its underlying theory by the AAFS or a similar body will be included as a consideration for admission as scientific evidence. This rationale has been used in the case of forensic disciplines such as handwriting analysis, and has resulted in expert testimony being nullified based on the lack of a scientific foundation [15].

Demonstrating the requirement of general acceptance is difficult even when concerns about the lack of a relevant scientific community are ignored. This has resulted in the default argument that practitioners use established vendor tools that are "industry standard" and the tools are, therefore, "generally accepted." The criteria governing "industry standard" are ambiguous at best. Often, an expert's choice of a tool is the outcome of an aggressive marketing campaign by a vendor; little or no direct testing or validation of the tool is conducted by the expert [11]. The cost of a tool rather than its scientific validity often impacts its general acceptance, especially since most law enforcement agencies have limited budgets.

5. Proposed Solution

No silver bullet exists for meeting all the F.R.E. 702 and Daubert considerations; therefore, interim approaches must be considered. As discussed above, no entity currently certifies computer forensic tools and no entity is accountable for their testing. Furthermore, no trusted third party currently attests to the reliability and validity of computer forensic tools. This is a logical area to start in developing a solution.

Numerous web sites created for electronic commerce applications implement a technology called secure socket layer (SSL) [3] to encrypt information in transit. Part of SSL revolves around the issuance and

maintenance of third-party certificates. SSL uses a trusted third party to verify information about the certificate holder and to ensure that the certificate provided matches that of the holder; this approach mitigates the risk of malicious access. The SSL model, based on a trusted third party, has gained wide acceptance in the electronic commerce community and has resulted in its ubiquitous use.

The computer forensics field could employ a trusted third party for certification purposes. Several companies and underwriting laboratories certify and accredit products, applications and hardware [12] (also see FIPS 140-2 [6]). Accounting entities have offered to certify the trustworthiness of websites and web transactions (e.g., WebTrust [1]). A logical extension would be for computer forensic tools (open source and proprietary) to be certified by impartial underwriters laboratories. By using this approach, intellectual property concerns of vendors can be alleviated and the blind faith reliance on vendors' assertions that their tools work as advertised can be set aside.

To be of any real value, the trusted organization must make both the results and its testing methodologies open to scrutiny (peer review). The end result of this process is a sort of "Good Housekeeping Seal" for computer forensic tools and an updated, publicly available list of approved tools that the courts could turn to for guidance on general acceptance and reliability.

The main limitation of this approach is liability, which will require the certifying entity to purchase liability insurance. To mitigate the problem of ever increasing malpractice insurance premiums as in the health care industry [2], a trusted third party who evaluates and certifies a computer forensic tool may offset some of the liability to the company that produced the tool. The third party would still have to carry liability insurance, but hopefully with reduced premiums.

Another limitation is the rate of change of computer forensic tools (new patches, versions and technologies). Often, vendors release new versions with minor changes every two to three months. This situation would require continuous re-testing, re-certification and re-publication of the test results, resulting in delays in the new version being released to the computer forensics community. Protracted delays have obvious economic ramifications to vendors and to practitioners, due to the inevitable price increases that would be passed to them by vendors. The certification of open source tools is an issue: Who will pay for certifying open source tools? Open source tools are often popular because they are free. Absent potentially costly certification, will the results obtained using open source tools be deemed inadmissible in court if only proprietary tools are certified?

The issue of whether or not the trusted third party should be a government agency or a private sector organization also must be considered. This is a contentious issue as there is the potential for a niche market or a monopoly by one company. Clearly, the certifying entity should be perceived as being completely impartial. The requirement of neutrality would tend to support the use of a government or quasi-government entity. Notwithstanding the private/public sector debate, trust is the key to the success of this model. If the computer forensic community distrusts the process, the model is flawed and the faith of the courts in the reliability and validity of the certification results will be undermined.

6. Conclusions

The number of court cases involving digital evidence will continue to increase as computers become more intertwined in society. Currently, the discipline of computer forensics and the derived digital evidence have difficulty meeting the F.R.E. 702 and Daubert considerations. This can have serious consequences for the computer forensics discipline as a whole. The discipline cannot survive for long if it relies on the lack of technical and scientific understanding by the courts. While U.S. courts have been willing to admit evidence generated by computer forensic tools based on face value, there is no guarantee that they will do so indefinitely [5]. As defense attorneys become more knowledgeable about computer forensics and digital evidence, there will be an increase in the number F.R.E. and Daubert challenges, more judicial scrutiny over what constitutes admissible digital evidence, more negation of testimony, and possibly increased suppression of evidence [10].

To minimize this potential, the computer forensics community must consider solutions that meet the Daubert considerations or risk the imposition of court-mandated solutions. Rather than attempting to reinvent the wheel, the community needs to look to other forensic sciences for direction and guidance, and, as suggested in this paper, adopt models and approaches that have proven to be effective. There is a real risk that if the computer forensics community does not act quickly and decisively, the discipline may end up being viewed by the courts as a pseudo science, or worse, a junk science.

References

[1] American Institute of Certified Public Accountants, WebTrust (www.cpawebtrust.org).

[2] Foundation for Taxpayers and Consumer Rights (www.consumerwatchdog.org/healthcare).

[3] A. Freier, P. Karlton and P. Kocher, *The SSL 3.0 Protocol* (wp.netscape.com/eng/ssl3/draft302.txt), 1996.

[4] E. Kenneally, Gatekeeping out of the box: Open source software as a mechanism to assess reliability for digital evidence, *Virginia Journal of Law and Technology*, vol. 6(13), 2001.

[5] O. Kerr, Computer crime and the coming revolution in criminal procedure, *Proceedings of the Cyber Crime and Digital Law Enforcement Conference*, 2004.

[6] NIST, *Security Requirements for Cryptographic Modules*, FIPS PUB 140-2 (csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf), 2001.

[7] NIST, *National Software Reference Library and Computer Forensics Tool Testing Project* (www.nsrl.nist.gov/Project), 2003.

[8] Ohio Court of Appeals, State of Ohio v. Brian Cook, 777 NE 2d 882, 2002.

[9] K. Poulsen, Microsoft: Closed source is more secure (www.securityfocus.com/news/191), 2001.

[10] F. Smith and R. Bace, *A Guide to Forensic Testimony: The Art and Practice of Presenting Testimony as an Expert Technical Witness*, Addison-Wesley, Boston, Massachusetts, 2003.

[11] Texas Appeals Court, Williford v. State of Texas, No. 11-02-00074-CR, 127 SW 3d 309, 312-313, 2004.

[12] Underwriters Laboratories (www.ul.com).

[13] U.S. Circuit Court of Appeals (DC Circuit), Frye v. United States, 293 F. 1013, 1923.

[14] U.S. Circuit Court of Appeals (11th Circuit), Four Seasons v. Consorcio, 267 F. Supp. 2d, 70, 2004.

[15] U.S. District Court (Alaska District), United States v. Saelee, 162 F. Supp. 2d 1097, 1105, 2001.

[16] U.S. Government, *Digital Millenium Copyright Act*, Pub. L. No. 105-304, 112 Stat. 2860 (www.copyright.gov/legislation/dmca.pdf), 1998.

[17] U.S. Government, *Federal Rules of Evidence* (judiciary.house.gov/media/pdfs/printers/108th/evid2004.pdf), 2004.

[18] U.S. Supreme Court, Daubert v. Merrell Dow Pharmaceuticals, 509 U.S. 579, no. 92-102, 1993.

[19] U.S. Supreme Court, Kumho Tire Company v. Carmichael, 526 U.S. 137, no. 97-1709, 1999.

[20] Washington Superior Court, State of Washington v. Leavell, Cause No. 00-1-0026-8, 1-17, 2000.

Chapter 5

NON-TECHNICAL MANIPULATION OF DIGITAL DATA

Legal, Ethical and Social Issues

Michael Losavio

Abstract This paper investigates basic issues related to the use of digital evidence in courts. In particular, it analyzes the basic legal test of authenticity of evidence with respect to an e-mail tool that can be used to manipulate evidence. The paper also examines the experiences and perceptions of U.S. state judicial officers regarding digital evidence, and reviews case law on how such evidence might be tested in the courts. Finally, it considers ethical and social issues raised by digital evidence and the mitigation of problems related to digital evidence.

Keywords: Digital evidence, e-mail evidence, authenticity

1. Introduction

Digital forensics bridges the science of computing and the judicial process. Both disciplines seek the truth, but their methods are distinct and their ends different. Every aspect of digital evidence, even the seemingly trivial, is tested during the judicial process. For example, issues related to the authenticity and integrity of e-mail messages are addressed during the administration of justice. Addressing these issues involves varying contributions by digital forensics with varying results.

This paper examines basic issues related to the use of digital evidence in courts and how digital forensics links computing to the judicial process. The growing use of digital evidence in judicial proceedings is discussed, focusing on the use of e-mail evidence in U.S. District Courts. The mutability and evanescence of electronic data raises issues concerning its authenticity that may lead courts to question its reliability as

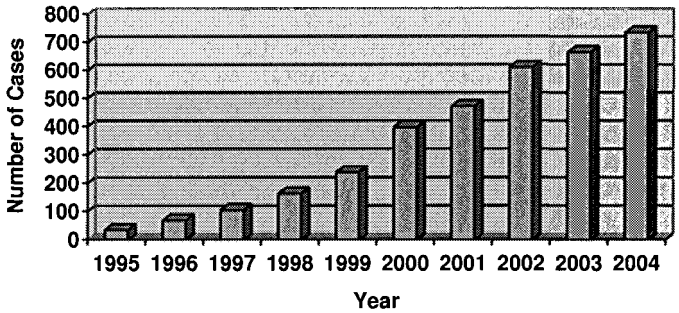


Figure 1. U.S. District Court cases referencing e-mail.

evidence. This issue of authenticity is investigated with respect to an e-mail tool that can be used to manipulate evidence.

The paper also examines the experiences and perceptions of U.S. state judicial officers regarding digital evidence, and reviews case law on how such evidence might be tested in the courts. Finally, it considers ethical and social issues raised by digital evidence and the mitigation of potential problems related to digital evidence.

2. Digital Evidence

Society's dependence on computers and networks assures the presentation of digital evidence in courts [11]. Electronic mail has become the very fabric of commercial litigation [20]. It creates a wealth of possible evidence and a growing "cottage industry" to assist litigators in discovering such evidence, with the legal commentary to explain it.

A keyword analysis of trial-level U.S. District Court opinions referencing e-mail shows a significant increasing trend over a ten-year period (Figure 1). The bulk of all cases in the United States, including most criminal cases and all domestic relations cases, are resolved by state courts whose opinions are not reported in legal databases. Nevertheless, anecdotal information suggests that the increasing trend in the use of e-mail evidence is common throughout the judicial system.

3. Relevance and Authenticity

Relevance and authenticity are two significant considerations that judicial officers must take into consideration before deciding whether or not to admit any evidence into court. Relevance is the truth or falsehood of a fact or issue in question; a test of relevance must not involve any undue prejudice against the opposing party. Authenticity, in its legal sense, means that something is what it is claimed to be. One of the

Table 1. Case data from a U.S. state court system.

Year	2001	2002	2003	2004
General jurisdiction cases terminated (by fiscal year (July 1–June 30))	115,800	117,900	129,600	138,500
Reported appellate opinions involving digital evidence (by calendar year)	2	3	3	3
Subset of reported appellate opinions involving challenges to admissibility	0	0	0	0

measures of authenticity is how evidence is demonstrated to be reliable. The legal testing of the authenticity of digital evidence has received less attention than techniques for digital forensic investigation and discovery [20]. In 2004, of about 163 reported U.S. federal appellate cases referencing e-mail, none addressed authenticity issues. Furthermore, of about 760 federal trial cases referencing e-mail, only four involved issues of authenticity of e-mail messages.

The ability to fabricate digital data makes authenticity a vital issue, even where some courts assert that digital mutability alone does not impact reliability [16, 25]. But this situation may change if authenticity challenges begin to exclude digital evidence from court proceedings. Robins [20] questions whether the computerized nature of digital evidence makes fabrication and/or errors more likely and, therefore, less reliable for decision-making.

Table 1 presents case closure statistics for trial courts, published appellate opinions involving digital evidence, and the admissibility of digital evidence for one U.S. state court system. Interestingly, from among more than 100,000 trial-level cases during each 12-month period, there were no more than three reported appeals in each period mentioning digital evidence, and not one case addressed admissibility or reliability. This may explain why there is so little case law guidance for individuals working in the discipline of digital forensics.

3.1 Legal Tests for Authenticity

In U.S. courts of law, the Rules of Evidence seek to assure the integrity and authenticity of evidence as a precondition for admissibility, to the end that the truth may be ascertained and proceedings justly determined [21, 24, 27, 28, 31, 32, 34–36]. In particular, *Federal Rule of Evidence*

901: Requirement of Authentication and Identification [34] and its state progeny provide as to any evidence, digital or otherwise:

> The requirement of authentication or identification as a condition precedent to admissibility is satisfied by evidence sufficient to support a finding that the matter in question is what its proponent claims.

This flexible rule permits authentication by direct testimony or analysis of contents, internal patterns or other distinctive characteristics. Digital evidence gets special treatment in this rule. In particular, the rule states that where "data are stored in a computer or similar device, any printout or other output readable by sight, shown to reflect the data accurately, is an original." Thus, a printout or other output of digital data may be used as the evidence [31]. Testimony pertaining to such a duplicate or copy of digital evidence is considered to be as reliable as the original (or "best evidence"). Evidence rules pertaining to hearsay also address reliability. In particular, out-of-court statements of third parties are not admissible absent showing such second-hand evidence is authentic and reliable, and need not withstand direct testing via cross-examination.

Robins [20] and Givens [11] suggest that these evidentiary rules might be liberally construed in a way that admits digital evidence with less rigor than non digital evidence. Givens [11] notes that some courts give greater credence to digital evidence because a computer is deemed less subject to error and manipulation than a human agent. However, several non-technical tools are available for manipulating digital information such as e-mail messages. As discussed below, these tools significantly increase the potential that digital evidence may be fabricated.

3.2 Fabrication of E-Mail Messages

A popular e-mail program was used to investigate the ease with which digital evidence could be fabricated. Although the fabrication may be obvious to experts, it may not be obvious to many individuals in the judicial system, including jurors, counsel and judges.

A digital forensics expert who reports on the fabrication or authenticity of digital evidence must be prepared to address what may be "obvious" to him/her in a clear, credible and non-condescending manner. Clarity and credibility may depend on truthful testimony that the expert has tested the "obvious" and has made hard findings. Speculation – even if it is scientifically grounded – may not be enough. The expert's answer to the question: *Have you ever tested this?* may be important to a judge or jury in accepting the expert's conclusions. In any case, testing is good scientific method.

Three tests were conducted using a popular e-mail program. The tests were used to create digital evidence, including potentially admissible printouts of e-mail messages.

Fabrication Test 1

The first test simply edits an existing e-mail message and prints out the edited version.

1 An existing e-mail message is opened using an e-mail program.

2 The e-mail message is edited by adding or deleting text.

3 The e-mail message is printed.

Examination of the printout of the fabricated e-mail message reveals that the edited version is indistinguishable on its face from the original.

Fabrication Test 2

Assuming a challenge to the paper "original," the e-mail program is used to fabricate the electronic copy of the e-mail message itself. In other words, the "best evidence" of the e-mail message is modified.

1 An existing e-mail message is opened using an e-mail program.

2 The e-mail message is edited by adding or deleting text.

3 The edited e-mail message is saved.

4 The e-mail program is closed.

5 The edited e-mail message is re-opened using the e-mail program, showing the edited text as part of the e-mail message.

The forged e-mail message is saved in digital form as the document itself. The content of the document cannot be demonstrated to be unreliable.

Fabrication Test 3

The `Properties` option in the e-mail program permits the review of the time an e-mail message was sent, received and last modified. This timestamp information may indicate tampering, raising questions about the authenticity and integrity of the "best evidence."

1 The system date and time are reset.

2 An existing e-mail message is opened using an e-mail program.

3 The e-mail message is edited by adding or deleting text.

4 The edited e-mail message is saved.

5 The e-mail program is closed.

6 The edited e-mail message is re-opened using the e-mail program, showing the edited text as part of the e-mail message.

7 The `Properties` option is executed, showing the reset date and time in the e-mail message timestamp.

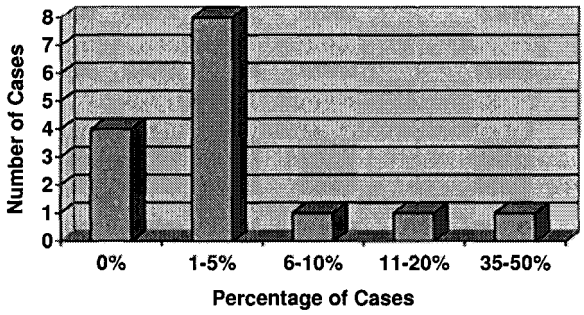


Figure 2. Caseloads of individual officers involving digital evidence.

Digital forensic analysis may be able to detect the modifications made to the e-mail message and the system date and time, but there are some hurdles. Can the costs of such analysis can be afforded by the parties? Is there sufficient skill or sufficient opportunity to use such analysis in every case? The judicial process will render judgment based on the evidence it is provided. How it does so depends, among other things, on judicial perceptions and experience with digital evidence.

4. Judicial Perceptions and Experience

A sample of state judicial officers were surveyed about their perceptions and experience with digital evidence and its reliability. These officers handle divorce, custody and maintenance actions. An examination of trial and appellate results from their jurisdictions indicates that there were more than 26,000 and 34,000 trial cases in 2001 and 2004, respectively [14]. No appellate opinions mentioned digital evidence [14].

4.1 Survey of State Judicial Officers

The survey results indicate that the majority of respondents had digital evidence in cases before them (Figure 2). In all, 75% of the respondents indicated that they had encountered digital evidence in proceedings. (Note that 52% of the individuals who were solicited responded to the survey.) Digital evidence appeared very frequently in cases before one respondent, but for others it was much less so. The frequency distribution suggests future growth in the use of digital evidence.

As shown in Figure 3, e-mail was the most common type of digital evidence in cases before 48% of the surveyed individuals (68% of the respondents), followed by web browsing items (30%/44%) and digital photos (26%/37%). The prevalence of e-mail evidence in domestic relations cases parallels Robins' observation for commercial litigation [20].

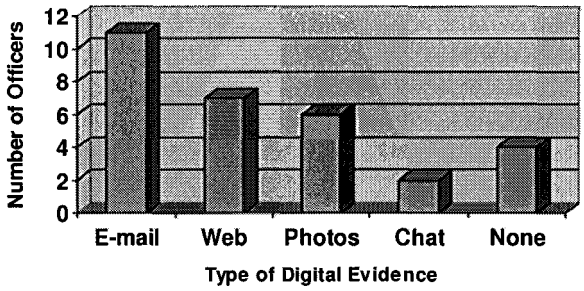


Figure 3. Judicial officers with cases involving digital evidence.

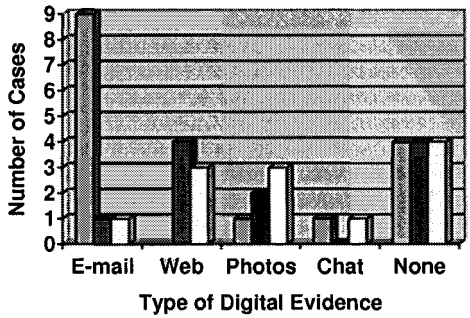


Figure 4. Frequency of various types of digital evidence.

Figure 4 shows that e-mail had the highest frequency of use among all types of digital evidence. 48% of the individuals (68% of respondents) had encountered e-mail evidence (68% of respondents). For 39% (56% of respondents), e-mail was the most frequently used digital evidence.

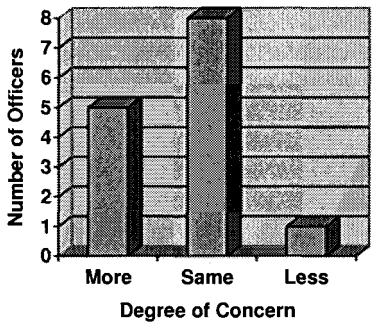


Figure 5. Comparison of concerns about evidence falsification.

The survey results also indicate that the majority of judicial officers had the same concerns about the falsification of digital evidence as non digital evidence. However, as shown in Figure 5, 22% (36% of

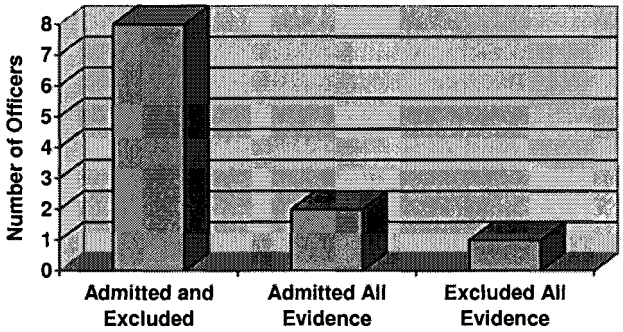


Figure 6. Judicial officers admitting and excluding digital evidence.

respondents) were more concerned about the possible falsification of digital evidence than non digital evidence. This also shows a difference in perception among some judicial officers about the reliability of digital evidence as opposed to traditional evidence.

4.2 Reliability Concerns

The survey results also indicate that the majority of judicial officers who were faced with digital evidence in cases had both admitted and excluded the evidence (Figure 6). They applied the state's rules of evidence, modeled on the U.S. Federal Rules, that test for relevance, undue prejudice, hearsay and authenticity in deciding whether to admit or exclude evidence [29, 30, 33, 34].

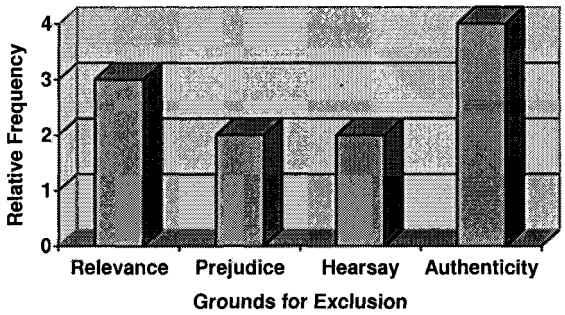


Figure 7. Grounds for excluding digital evidence.

Figure 7 shows that the lack of authenticity was the most frequent reason for excluding digital evidence. Hearsay, another reliability filter, was also noted as a ground for exclusion. Relevance and undue prejudice were also grounds for excluding evidence, but decisions regarding them are often made prior to determining authenticity. As a result, evidence

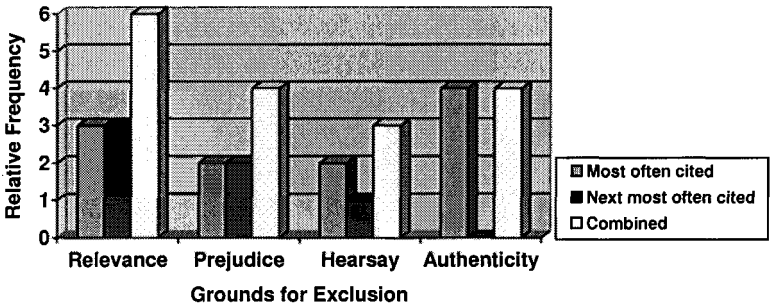


Figure 8. Most common grounds for excluding evidence.

excluded for reasons of hearsay or prejudice are usually not tested for authenticity.

Figure 8 shows the relative frequencies of the most often cited and the next most often cited grounds for excluding digital evidence. Combining the values for the most often and next most often grounds for exclusion show that relevance and undue prejudice play significant roles in excluding digital evidence in domestic relations proceedings. Some of these cases may have benefited from digital forensic analysis, but others did not.

The default response, absent digital forensic analysis, is that traditional factors are used to decide authenticity, such as credibility and circumstantial evidence. Thus the author of an e-mail message can be authenticated by distinctive characteristics and the circumstances surrounding the e-mail message as well as by circumstantial evidence of the author's style and personal information he/she would have known [17, 23, 27].

Does the easy fabrication of e-mail evidence, coupled with the lack of digital forensic expertise and the use of non-technical, circumstantial evidence, by default, raise questions about the fairness and reliability of digital evidence? Should a domestic violence case involving e-mail threats be dismissed due to the technical superiority of one party over another, e.g., Rabic v. Constantino [8]? Does this raise ethical and social issues that must be addressed by the digital forensics discipline?

5. Ethical and Social Concerns

The easy fabrication of digital evidence and the existence of a "digital divide" (between those who can afford digital forensic services and those who cannot go) beyond the courts. Unless these issues are addressed, the confidence of any process that relies on digital evidence can be undermined. Natsui [18] notes several due process (basic fairness)

issues concerning information security and electronic personal information (both of which involve digital forensics). Furthermore, while OECD Guidelines hold that information security should be compatible with the fundamental values of a democratic society, they do not explicitly recognize any due process rights [18, 19]. For attorneys, their obligations relating to competence and integrity in working with opposing parties place a professional onus on them to assure that digital evidence is used properly [2–6]. Lack of knowledge may not be a valid defense to ethical and other sanctions in cases where an attorney, exercising diligence in investigating the merits of the case, discovers falsified evidence, e.g., Jimenez v. Madison Area Technical College [22, 26].

The obligations are more diffuse for experts in digital forensics. Expert witnesses in litigation are generally protected under the witness immunity doctrine so long as they do not perjure themselves. However, this immunity is under scrutiny and the possibility exists that expert witnesses could be held to some standards for liability [12, 13]. Such standards might be applied to digital forensic experts as their work moves from laboratory to courtroom, but they will probably be applied first to experts in licensed, regulated professions, e.g., medical practice. To the extent that digital forensics falls with the computing discipline, it does not yet have fully realized practices and procedures requiring responsible engineering and science even though its professional societies (ACM and IEEE) aspire to such [1, 10, 13]. Indeed, Linderman [15] contends that, unlike medical practice, information technology is not a "profession" with attendant, articulated obligations. On the other hand, Denning [9] argues that guidance is needed in the information technology discipline as public trust and confidence, once lost, may take years to rebuild. Without guidance as to what should and should not be done in digital forensics, there is a risk that missteps might be taken that would affect public confidence in the discipline. Clearly, the digital forensics discipline must take greater responsibility for the veracity of digital information. While this is not a perfect solution, it is critical to probative processes in the digital age.

6. Conclusions

The use of digital evidence is growing in American courts. Meanwhile, non-technical tools are becoming available that make it very easy to fabricate digital evidence. At present there is little case law guidance regarding the reliability of digital evidence in court, and some trial judges are concerned that digital evidence offers greater opportunities for falsification than traditional evidence. The discipline of digital forensics

does not currently offer a complete response to those concerns. Judges and lawyers may not be fully versed in the use of digital evidence. Parties may not be able to afford digital forensic expertise, or one party may be able to "outgun" another with experts. And there is still the issue of certification of digital forensics experts, or some other way to validate the competence of purported experts in what is growing area of analysis. Digital forensics must address technology as well as perceptions of reliability and fairness to bridge the gap between computing and judicial processes. Failure to address these matters will hurt the truth-finding enterprise.

Aristotle [7] noted the differences in how the scientific and political disciplines pursue truth, and he advised not to expect more than is possible. The critical tasks are to reduce uncertainty, both intended and unintended, and to promote fairness and truth-finding processes with regard to digital information in every forum.

References

[1] ACM, ACM Code of Ethics and Professional Conduct (www.acm.org/constitution/code.html), October 16, 1992.

[2] American Bar Association, Model Rule 1.1: Competence (www.abanet.org/cpr/mrpc/mrpc_toc.html).

[3] American Bar Association, Model Rule 3.1: Meritorious Claims and Contentions (www.abanet.org/cpr/mrpc/mrpc_toc.html).

[4] American Bar Association, Model Rule 3.3: Candor toward the Tribunal (www.abanet.org/cpr/mrpc/mrpc_toc.html).

[5] American Bar Association, Model Rule 3.4: Fairness to Opposing Party and Counsel (www.abanet.org/cpr/mrpc/mrpc_toc.html).

[6] American Bar Association, Model Rule 3.8: Special Responsibilities of a Prosecutor (www.abanet.org/cpr/mrpc/mrpc_toc.html).

[7] Aristotle, *Nicomachean Ethics*, W. Ross (Translator), Oxford University Press, New York, 1998.

[8] California Court of Appeals (First District, Division Four), Rabic v. Constantino, Case No. A106248, 2004.

[9] P. Denning, Who are we? *Communications of the ACM*, vol. 44(2), pp. 15-19, 2001.

[10] P. Denning, Crossing the chasm, *Communications of the ACM*, vol. 44(4), pp. 21-25, 2001.

[11] J. Givens, Comment: The admissibility of electronic evidence at trial: Courtroom admissibility standards, *Cumberland Law Review*, vol. 34, pp. 95-126, 2003.

[12] J. Harrison, Reconceptualizing the expert witness: Social costs, current controls and proposed responses, *Yale Journal on Regulation*, vol. 18(2), pp. 253-314, 2001.

[13] IEEE, IEEE Code of Ethics (onlineethics.org/codes/IEEEcode.html).

[14] Kentucky Court of Justice, FY 2004 Historical Reports: Circuit Court Data and Circuit Family Data, Domestic and Family Case Closures (www.kycourts.net/AOC/CSRS/ResearchStats.shtm).

[15] J. Linderman and W. Schiano, Information ethics in a responsibility vacuum, *The DATA BASE for Advances in Information Systems*, vol. 32(1), pp 70-74, 2001.

[16] M. Losavio, The secret life of electronic documents, *Kentucky Bench & Bar*, vol. 63(5), pp. 31-34, 1999.

[17] Maine Supreme Court, State v. Turner, 2001 ME 44 (www.courts.state.me.us/opinions/documents/01me44tu.htm), 2001.

[18] T. Natsui, Procedural justice: Changes in social structures in an information society and the maintenance of justice, presented at the *International Conference on the Internet and the Law – A Global Conversation*, 2004.

[19] Organisation for Economic Cooperation and Development, Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security (www.oecd.org/dataoecd/16/22/15582260.pdf).

[20] M. Robins, Evidence at the electronic frontier: Introducing e-mail at trial in commercial litigation, *Rutgers Computer & Technology Law Journal* vol. 29, pp. 219-315, 2003.

[21] W. Stallings, *Network Security Essentials (2nd Edition)*, Prentice-Hall, Upper Saddle River, New Jersey, 2003.

[22] U.S. Circuit Court of Appeals (7th Circuit), Jimenez v. Madison Area Technical College, 321 F.3d 652, 2003.

[23] U.S. Circuit Court of Appeals (11th Circuit), United States v. Siddiqui, 235 F.3d 1318, 2000.

[24] U.S. Department of Justice, Part V: Evidence, *Manual for Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations*, Computer Crime and Intellectual Property Section, Criminal Division, Washington, D.C., 2002.

[25] U.S. District Court (Illinois Northern District, Eastern Division), Tibbetts v. RadioShack Corporation, Case No. 03 C 2249, 2004.

[26] U.S. Government, *Federal Rules of Civil Procedure*, U.S. Government Printing Office, Washington, D.C., 2004.

[27] U.S. Government, Rule 101 (Scope), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., p. 1, 2004.

[28] U.S. Government, Rule 102 (Construction and purpose), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., p. 1, 2004.

[29] U.S. Government, Rule 402 (Relevant evidence generally admissible; irrelevant evidence inadmissible), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., p. 3, 2004.

[30] U.S. Government, Rule 403 (Exclusion of relevant evidence on grounds of prejudice, confusion or waste of time), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., p. 3, 2004.

[31] U.S. Government, Rule 702 (Testimony by experts), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., p. 13, 2004.

[32] U.S. Government, Article VII: Rules 801-807 (Hearsay), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., pp. 15-20, 2004.

[33] U.S. Government, Rule 802 (Hearsay rule), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., p. 15, 2004.

[34] U.S. Government, Rule 901 (Requirement of authentication or identification), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., pp. 20-21, 2004.

[35] U.S. Government, Rule 1001 (Definitions), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., pp. 23-24, 2004.

[36] U.S. Government, Article X: Rules 1001-1007 (Contents of writings, recordings and photographs), *Federal Rules of Evidence*, U.S. Government Printing Office, Washington, D.C., pp. 23-24, 2004.

II

INVESTIGATIVE TECHNIQUES

Chapter 6

DETECTING SOCIAL ENGINEERING

Michael Hoeschele and Marcus Rogers

Abstract This paper focuses on detecting social engineering attacks perpetrated over phone lines. Current methods for dealing with social engineering attacks rely on security policies and employee training, which fail because the root of the problem, people, are still involved. Our solution relies on computer systems to analyze phone conversations in real time and determine if the caller is deceiving the receiver. The technologies employed in the proposed Social Engineering Defense Architecture (SEDA) are in the proof-of-concept phase but are, nevertheless, tractable. An important byproduct of this work is the generation of real-time signatures, which can be used in forensic investigations.

Keywords: Social engineering, defense architecture, attack signatures, forensics

1. Introduction

Much of the research literature on social engineering focuses on its use in perpetrating computer-related crime as opposed to detecting or preventing social engineering attacks. One exception is the work by Rogers and Berti [15] that discusses how to prevent social engineering attacks. Most of their suggestions are related to security policies and training personnel to be aware of what social attacks may look like; both of these are dependent on the human element. Dolan [4] makes similar suggestions, claiming that successful social engineering attacks rely on the employees of an organization. He emphasizes that to contain such attacks, employees must be well-trained and familiar with common social engineering techniques.

Social engineering is a clear threat. In a 1997 article, Bort [3] noted that: "Of the 384 respondents who confessed to being attacked over the last year, social engineering was to blame in 15 percent of the cases – the second largest cause." Other more recent publications, such as the

annual FBI/CSI Computer Crime and Security Surveys [5, 12, 14], do not consider social engineering attacks *per se*. But the survey results and accompanying discussion allude to social engineering being an ever present threat.

According to Bort [3], no hardware or software can defend information systems against a human being telling a convincing lie. While this may still be true, natural language processing researchers focusing on the problem of lie detection have made some progress. In particular, Raskin and co-workers [13] have proved that the problem is tractable and have created a working model.

This paper considers the problem of detecting social engineering attacks perpetrated over phone lines. The solution relies on computer systems to analyze phone conversations in real time and determine if the caller is deceiving the receiver. While the technologies employed in the proposed Social Engineering Defense Architecture (SEDA) are still in the proof-of-concept phase, they are, nevertheless, tractable [13]. In any case, the attack signatures generated as a consequence of this work can be used in forensic investigations of social engineering attacks.

2. Problem Statement

Social engineering attacks are a threat to all organizations. However, forensic investigations principally focus on attacks perpetrated using computer systems. Little, if any, work has considered the forensic analysis of social engineering attacks. Signatures have not been identified for social engineering attacks nor have systems been developed to log activity associated with such attacks. This means that even if it can be determined that a social engineering attack has occurred, it is very unlikely that the crime can be traced back to the perpetrator, let alone be prosecuted. Social engineering attack signatures are somewhat elusive because of the nature of the attacks and their avenue, which is most commonly the telephone system. Other avenues for attack, such as face-to-face conversations, are quite rare because of the risk incurred by attackers. This work considers the problem detecting of social engineering attacks perpetrated over phone lines. The proposed Social Engineering Defense Architecture (SEDA) can generate attack signatures in real time. These signatures allow a logging facility to also serve as a defense mechanism as in a typical network intrusion detection system.

3. Social Engineering

Dolan [4] defines social engineering as "using relationships with people to attain a goal." For the purposes of this work, the term social engineering is discussed with reference to attackers who attempt to illegally compromise an organization's assets. It should be noted that the types of organizations under attack are not limited to faceless multinational corporations. Educational institutions, banks and even the corner video store are equally at risk.

3.1 Methods

Social engineers, like computer hackers, take advantage of system weaknesses. As Dolan [4] states:

> "*Social engineers use tactics to leverage trust, helpfulness, easily attainable information, knowledge of internal processes, authority, technology and any combination there of. They often use several small attacks to put them in the position to reach their final goal. Social engineering is all about taking advantage of others to gather information and infiltrate an attack. The information gained in a phone book may lead to a phone call. The information gained in the phone call may lead to another phone call. A social engineer builds on each tidbit of information he or she gains to eventually stage a final, deadly attack. A successful social engineering attempt could result in great financial loss for the target company. A motivated attacker will be willing to gain information in any way possible.*"

Social engineering is successful because people, in general, have a desire to help others and gain satisfaction from it [4]. An expert social engineer has the ability to establish trust and usually masquerades as someone the victim would trust.

Much of the information necessary to perpetrate social engineering attacks is publicly available. Reverse phone look-up directories, such as www.reversephonedirectory.com, are freely available on the Internet. Once a phone number and address are obtained, other useful information can be obtained effortlessly. The web pages of many organizations hold considerable information, e.g., organizational charts and biographical sketches, that can be used in social engineering attacks.

One common social engineering technique is to call the main switchboard of an organization and ask to be transferred to an employee. The receiver of the transfer call does not typically have a phone number that is posted publicly, so he assumes that the call is from an insider. This can prove to be a strong enough credential to allow more internal numbers to leak out, furthering the social engineer's cache of information. Arthurs [2] identifies other examples of social engineering attacks:

- **IT Support:** A social engineer claiming to be from the company's IT support group phones a user and explains that he is locating faults in the company network. He has narrowed the fault to the user's department but he needs a user ID and password from a department employee to identify the problem. Unless the user has been properly educated in security practices, he will very likely give the "trouble-shooter" the requested information.
- **Manager:** A social engineer, using a perceived position of authority, phones the help desk demanding to know why he cannot login with his password. He intimidates the help desk into giving him a new password by emphasizing that he has only a limited time to retrieve information for a report to the company vice president. He may also threaten to report the help desk employee to his supervisor.
- **Trusted Third Party:** A social engineer phones the help desk claiming to be the vice-president's executive assistant. She says that the vice-president has authorized her to collect the information. If the help desk employee balks, she threatens to inform the employee's supervisor.

It can be seen from these examples that the majority of social engineering attacks are committed over the telephone and rely on the fact that the receiver of the call takes the caller's word about his/her identity. Typically, there is no authentication other than answering questions pertaining to information that only an employee would know.

3.2 Motives

The Hackers Manifesto [10] explains why hackers desire to break into secure systems. The primary drivers are the quest for knowledge and the challenge. While harm may not be the intent, it is clear that considerable damage can be caused.

Much more dangerous are social engineering attacks that are intended to compromise an organization's assets. Examples are a recently fired employee seeking vengeance and a seasoned social engineer attempting to steal corporate information. These individuals have a better chance of succeeding and causing damage as they have greater motivation to succeed and potentially more resources at their disposal [4].

3.3 Targets

The targets of social engineering attacks range from personal information to intellectual property. However, it is important to note that

these are only the end targets of the attack, and many small pieces of information must be obtained before the final target can be reached. The information includes organization policies, protocols, hierarchy, phone books and server names. Much of this information is available on an organization's website or is obtained by making phone calls to employees. The real challenge is protecting such seemingly trivial information without interfering with day-to-day operations [15].

4. Current Solutions

Security policies and employee training are the two main approaches for preventing social engineering attacks. However, these approaches are fundamentally flawed as they rely on humans to patch security holes. The flaw is that human trust is the vulnerability that is exploited by social engineers; the notion of trust is deeply embedded in Western culture and is difficult to overcome [15]. A good social engineer can be very convincing that he or she needs the requested data, and that the individual who receives the request is hurting the organization by not helping. Employee training may stop novice attackers, but a seasoned social engineer will likely not be considered a risk during a conversation, so the training will never be triggered. This point is evident in the social engineering attacks described in the previous section (see [15] for additional details).

4.1 Security Policies

Security policies alone cannot prevent break-ins. Often, a security policy is effective only in the sense that after the policy is broken, it is easy to display the policy and show how it was violated. However, a security policy that classifies data into different levels of sensitivity can be quite effective as it requires a social engineer to obtain higher credentials to gain access to sensitive information. The end result is more work, which would deter most casual social engineers. However, an experienced, persistent social engineer will likely keep working until he has all the credentials needed to access the information.

Allen [1] advocates security policies because they provide clear direction on what is expected of employees in an organization. Equally important is limiting data leakage by reducing the amount of specific data that is available. This has the effect of making social engineering attacks arduous and time consuming.

4.2 Employee Training

Employee training is currently the most effective deterrent to social engineering attacks. The training ranges from annual multi-day seminars to constant reminders via posters and mailings. The idea is that if employees know how social engineers execute attacks and gain trust, they will be able to detect attacks as they occur and take steps to defeat them. Employees are also encouraged not to release certain information over the phone, e.g., passwords and ID numbers. But the problem is that an expert social engineer never shows any signs of being an attacker; often, the social engineer appears to be a very conscientious employee. Therefore, it is not realistic to expect employees to be the primary defense against social engineering attacks. It is, however, logical to make them aware of social engineering attacks. This also helps in gaining employee acceptance of policies and systems designed to defend against social engineering attacks [15].

4.3 Evaluation of Current Solutions

Unfortunately, it is impossible with the available data to determine the effectiveness of current methods for dealing with social engineering attacks. It could be reasoned that the general lack of data shows the inadequacy of current methods of detection. For example, if network intrusion detection systems existed, how would one measure the number of attacks? It is clear that social engineering poses serious security threats, but no metrics exist for measuring its impact. Therefore, in addition to enhancing existing solutions and developing new prevention and detection techniques, research efforts must focus on measuring the impact of social engineering attacks.

5. Proposed Solution

This section describes the Social Engineering Defense Architecture (SEDA). SEDA is intended to detect social engineering attacks perpetrated over telephones and to generate logs for forensic investigations. The focus on the telephone medium is crucial as most social engineering attacks are carried out over the phone [4, 6].

SEDA is designed to detect attacks based on intent and deception instead of the attack target. Detecting a social engineering attack based on its target is difficult because social engineers typically first pursue targets with seemingly very little importance as discussed in Section 3. However, this trivial information is then used to obtain more sensitive

and well-guarded information. By detecting lying and deception, SEDA will help prevent social engineering attacks in their early and late stages.

5.1 Attack Detection

The primary purpose of SEDA is to make recipients of phone calls within an organization aware of callers who are attempting to deceive them or obtain unauthorized information. The "muscle" of the system is a text-independent voice signature authentication system. Markowitz defines text-independent verification as "[accepting] any spoken input, making it possible to design unobtrusive, even invisible, verification applications that examine the ongoing speech of an individual" [9]. According to Markowitz, the ability of text-independent technology to operate unobtrusively in the background makes it attractive for customer-related applications, because customers need not pause for security checks before moving on to their primary business objectives. The result is a system of authentication that hinders workflow marginally, if at all.

The voice signatures collected by SEDA will be linked to a database of personal information that includes the employee name, corporate association, job title, and all the phone numbers used to place calls. The types and amount of information gathered would depend on the needs of the organization employing SEDA.

The success of SEDA's strategy lies in the fact that social engineers often masquerade as employees to gain trust. The system would prevent social engineers from claiming to be employees, even if they have all the information to pass as one in a phone conversation. It would also complicate matters for a social engineer who keeps changing his name. The first time the attacker calls, the name he uses is associated with his voice signature; this would require him to modify his voice if he calls again under another name. While this is a way to defeat SEDA, most attackers would be deterred. In any case, attackers who modify their voices would still have to deal with SEDA's other attack detection systems.

The SEDA design also incorporates a voice-to-text engine, which can convert voice conversations into text in real time. Several prototype voice-to-text systems have been developed (see, e.g., [7, 8]). It is important that the voice-to-text conversion be performed rapidly and accurately. If the generated text cannot be sent for analysis fast enough, the attacker could obtain the requested information before the recipient of the call can be notified that an attack is in progress. Also, the voice-to-text conversion must be robust enough to deal with bad phone connections.

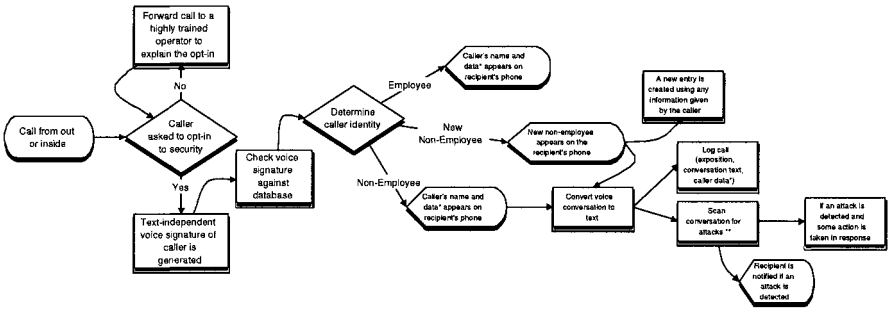


Figure 1. SEDA decision tree.

To support forensic investigations, all conversations originating from outside the company's phone switch should be recorded and the text of the conversations should be linked to the callers' and receivers' voice signatures. Because of the need to record conversations for security purposes, each caller would have to opt-in by calling a specific number when he/she first calls. This removes the expectation of privacy associated with telephone conversations, and ensures that the use of SEDA does not violate wire tap laws [16]. Otherwise, the caller would be transferred to an operator who is trained to resist social engineering attacks; this operator would explain the purpose of the opt-in process to the caller.

A textual conversation analysis tool is the "brain" behind SEDA. Raskin and co-workers [13] have developed a content analysis tool that uses sophisticated natural language processing techniques to determine if a person is lying. While the tool is not yet ready to be incorporated within SEDA, the research shows that the problem of parsing a conversation and determining if someone is lying is tractable. Due to its computational needs, such a tool will have to run on multiple servers to analyze conversations in real time. Nevertheless, it may only be a matter of time before conversation analysis tools are used in SEDA and other applications [13].

A simpler content analysis tool with a narrower scope could monitor conversations for specific strings used in social engineering attacks. These strings are similar to virus signatures. For example, if a caller says, "Please read me your username and password," it is clear that either the caller has malicious intent or he is violating the security policy, neither of which is acceptable. These rules would have to be customized for each organization. Figure 1 presents SEDA's decision tree structure. An expanded view of the attack detection process is shown in Figure 2. It should be noted that no action, aside from notifying the receiver of the call, is taken when an attack is detected. Further research is needed

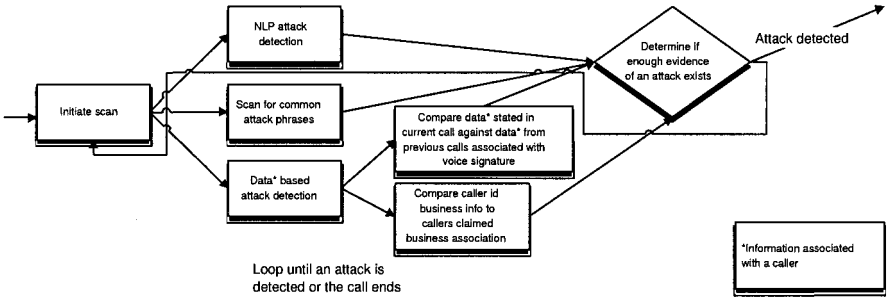


Figure 2. Attack detection process.

to determine the best course of action when a social engineering attack is detected.

5.2 Attack Signature Generation

As with attacks on computer systems, some social engineering attacks will get through no matter what is done to prevent them. In the case of a skilled social engineer breaking into an organization, SEDA provides call logs to perform a forensic analysis of the attack. As noted above, every conversation originating outside the company phone switch is recorded in text format with the voice signatures linked for caller identification. Storing conversations in text format reduces storage needs and permits scanning for clues without having to convert voice into text. Logs of conversations provide forensic investigators with information to trace criminal activity from the final target to the attacker. The logs could also be used in conjunction with other forensic methods to identify attackers.

6. Limitations

One of the major limitations of the proposed solution is its inability to deal with voice modulation. If an attacker were to mask his voice using a modulation device during every call in an attack, the ability to link the calls during a forensic investigation would be greatly decreased. However, voice modulation would have no effect on SEDA's ability to detect deception based on conversation content. Even a resourceful social engineer would not be able to bypass all of the SEDA's levels of protection.

Problems could also arise if SEDA is unable to handle poor telephone connections, e.g., from a cell phone. This situation can be viewed as another form of voice modulation.

A related problem is a replay attack – recording someone's voice and playing it into the phone to obtain authorization, and then continuing

the conversation. Text-independent voice signatures can be generated over the entire call to deal with such attacks. Many commercial speaker-verification systems look for telltale auditory signals, distortions, exact matches, and other indications that a recording has been used [9]. In fact, some voice signature systems already detect such attacks [9].

7. Future Research

This work suggests several avenues for future research. One open problem is handling internal calls. Our solution treats all calls the same regardless of the location of the caller. This could be problematic when there are a large number of internal calls. A technique for streamlining internal calls would significantly reduce the computational requirements.

As mentioned earlier, research is also needed to determine how to handle social engineering attacks after they are detected. Strategies for dealing with attacks would depend on the security goals of the organization. Nevertheless, guidelines for responding to attacks must be created to assist in the development of countermeasures.

Another area for future research is social engineering forensics. Forentic tools must be developed to parse log files and discover clues in logs generated by SEDA. In addition, policies for conducting forensic investigations using SEDA logs must be created. Essentially, much of the research in digital forensics has to be applied to investigating social engineering attacks.

8. Conclusions

The principal advantage of SEDA is that it takes the human element out of determining a person's identity over the phone. Callers will be identified as employees or outsiders; this alone is crucial to preventing social engineering attacks. The ability to detect deception also means that a social engineer will not be able to appeal to someone's emotions or try to bully him or her into performing an action. Another advantage is that the log files that are generated could support forensic investigations of social engineering attacks, which is an interesting new area of research in digital forensics. The main weakness is that voice modulation makes it possible for one person to call many times under different names and not be tracked. While this is a problem that must be addressed by efforts in voice modulation detection, it does not undermine SEDA. Despite the limitations, SEDA addresses two major problems that are so far unanswered: how to detect social engineering attacks and how to perform forensic analyses of social engineering attacks.

References

[1] M. Allen, The use of social engineering as a means of violating computer systems (www.sans.org/rr/catindex.php?cat_id=51).

[2] W. Arthurs, A proactive defense to social engineering (www.sans.org /rr/catindex.php?cat_id=51).

[3] J. Bort, Liar, Liar, *Client Server Computing*, vol. 4(5), 1997.

[4] A. Dolan, Social engineering (www.sans.org/rr/catindex.php?cat_id =51).

[5] L. Gordon, M. Loeb, W. Lucyshyn and R. Richardson, *2004 CSI/FBI Computer Crime and Security Survey* (www.gocsi.com), 2004.

[6] D. Gragg, A multilevel defense against social engineering (www.sans. org/rr/catindex.php?cat_id=51).

[7] C. Karat, C. Halverson, D. Horn and J. Karat, Patterns of entry and correction in large vocabulary continuous speech recognition systems, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 568-575, 1999.

[8] J. Lai and J. Vergo, MedSpeak: Report creation with continuous speech recognition, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 431-438, 1997.

[9] J. Markowitz, Voice biometrics, *Communications of the ACM*, vol. 43(9), pp. 66-73, 2000.

[10] The Mentor, *The Hackers Manifesto* (www.geocities.com/Silicon Valley/Heights/1926/mentor.html).

[11] Nemesysco, The Layer Voice Analysis (LVA) technology (www. nemesysco.com/technology-lvavoiceanalysis.html).

[12] R. Power, *2002 CSI/FBI Computer Crime and Security Survey* (www.gocsi.com), 2002.

[13] V. Raskin, F. Christian and K. Triezenberg, Semantic forensics: An application of ontological semantics to information assurance, *Proceedings of the Forty-Second Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 2004.

[14] R. Richardson, *2003 CSI/FBI Computer Crime and Security Survey* (www.gocsi.com), 2003.

[15] M. Rogers and J. Berti, The forgotten risk, in *Information Security Management Handbook, Volume 3*, H. Tipton and M. Krause (Eds.), CRC Press, New York, pp. 51-63, 2002.

[16] U.S. Department of Justice, 18 U.S.C. 2511 – Interception and disclosure of wire, oral or electronic communications prohibited (www.cybercrime.gov/usc2511.htm).

Chapter 7

A FRAMEWORK FOR EMAIL INVESTIGATIONS

Automated Information Extraction and Linkage Discovery

Anthony Persaud and Yong Guan

Abstract Email communications are commonly used by criminal entities to perpetrate illegal activities such as fraud and phishing scams as well as for transmitting threats and viruses. Due to the complexity of the task, it is often difficult for investigators to manually analyze email-related evidence. Automated techniques for email analysis can significantly enhance computer crime investigations. This paper proposes a framework for email investigations that incorporates automated techniques for information extraction and linkage discovery. The application of text/data mining and link analysis techniques assists in inferring social networks and in accurately correlating events and activities from email-related evidence.

Keywords: Email investigations, linkage analysis, information extraction

1. Introduction

Numerous crimes are committed using email communications. Emails are commonly used to perpetrate phishing scams as well as to transmit threats and viruses. In many cases, email communications provide evidence of conspiracy, helping identify new suspects and linking them to specific criminal activities. For example, in the Enron scandal, investigations of email correspondence showed that several top executives conspired to commit fraud and money laundering [6]. More recently, email evidence suggested that Merck executives may have known about the deadly side-effects of Vioxx since March 2000, long before it was removed from store shelves [3, 11].

Due to the complexity of the task, it is often difficult for investigators to manually analyze email-related evidence. Automated techniques for email analysis can significantly enhance computer crime investigations. This paper proposes a framework for email investigations that incorporates automated techniques for information extraction and linkage discovery. The application of text/data mining and link analysis techniques assists in inferring social networks and in accurately correlating events and activities from email-related evidence.

2. Problem Definition

Email messages comprise header information and a message body. Information in email messages ranges from partially structured information to unstructured information.

Simple Mail Transfer Protocol (SMTP) headers are examples of fixed information fields in email messages that provide formatted information on routing, time, date and addresses. Email message bodies contain unstructured information because no regulations are specified for message content. For example, a message can be written in a language other than English, or it may contain undefined acronyms, or it may use different writing styles and not have correct punctuation.

Analyzing email header information and the frequency of messages can provide insights into the communication patterns of individuals. The message body is essential to understanding the context of these patterns. Manual analysis of email-related evidence is arduous and time consuming. Unfortunately, the mixture of structured and unstructured information in email messages makes it difficult to create a fully automated process for analysis. Therefore, the main goal of this work is to provide a framework for automating information extraction and analysis during investigations of email-related evidence.

3. Related Work

Link discovery encompasses a broad range of topics such as discovering social networks, analyzing fraudulent behavior, detecting preemptive threats and modeling group activities. The InFlow organizational network analysis tool constructs social maps from email messages [8]. InFlow uses the `To` and `From` header fields and the frequency of emails sent between individuals to create social maps of organizations.

Other research uses email analysis to understand user behavior. Boyd and Potter [2] created Social Network Fragments as a self-awareness application for digital identity management. The system uses address fields from user email files to construct a social behavior map. For

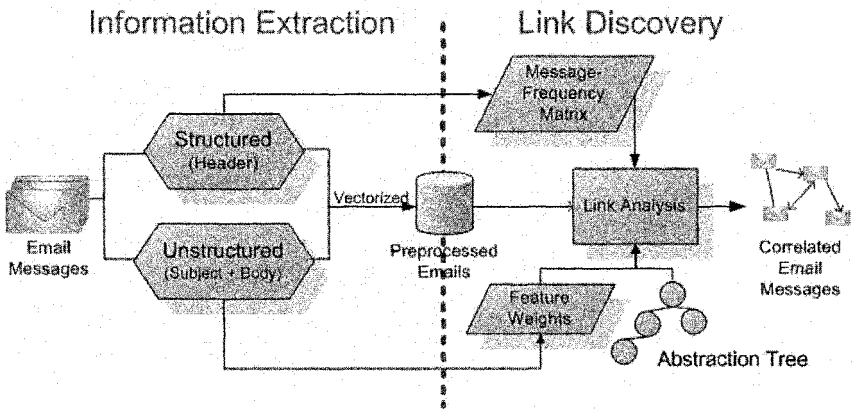


Figure 1. Email analysis framework.

example, the system helps identify a user's interests during a certain time period by analyzing his/her mailing list subscriptions.

4. Email Investigation Framework

Figure 1 presents an overview of the proposed email investigation framework. The framework involves two phases: information extraction and link discovery.

The information extraction phase involves structured information extraction and unstructured information extraction, which condense and summarize email messages using a feature vector format. Additionally, as described below, the processes produce a message frequency matrix and a set of feature weights.

The link discovery phase analyzes the vector-formatted email files, message frequency matrix and the corresponding feature weights, producing correlated pairs that manifest hidden relationships between communicating parties.

5. Information Extraction

This section describes the techniques used to extract structured and unstructured information.

5.1 Structured Information Extraction

The main goal of structured information extraction is to build messaging relationships. Therefore, the focus is on extracting information contained in the `To`, `From`, `Cc`, `Bcc`, `Reply-To` and `Delivered-To` address

Table 1. Message frequency matrix.

	$addr_1$	$addr_2$	...	$addr_{n-1}$	$addr_n$
$addr_1$	0	f_{12}	...	$f_{1(n-1)}$	f_{1n}
$addr_2$	f_{21}	0	...	$f_{2(n-1)}$	f_{2n}
...	...	...	...	...	...
$addr_{n-1}$	$f_{(n-1)1}$	$f_{(n-1)2}$	...	0	$f_{(n-1)n}$
$addr_n$	f_{n1}	f_{n2}	...	$f_{n(n-1)}$	0

fields. The address fields in SMTP headers are described in RFC 821 [9]. The format of email addresses is defined by RFC 2822 [10].

Address information is stored in a message frequency matrix that records the frequency of communication between pairs of email addresses. A sample message frequency matrix is presented in Table 1, where $addr_x$ is the primary email address of individual x and f_{xy} is the frequency of messages sent by $addr_x$ and received by $addr_y$. Note that `Received-By` is used instead of `Sent-To` because multiple individuals may have received an email message based on addresses in the `Cc` and `Bcc` fields.

Two rules are followed when collecting email addresses. First, all known email addresses used by a sender are recorded as a single (primary) address. For example, if an individual uses `me@email.com` and `myself@email.com`, one of the addresses, say `me@email.com`, is designated as the primary address. It does not matter which address is chosen as long as frequencies are recorded for a single address.

The second rule is that mailing list addresses, when known, are expanded to their corresponding membership addresses. Each membership address should be recorded in the message frequency matrix. For example, suppose mailing list `mlist@email.com` contains the addresses `them@email.com`, `him@email.com` and `her@email.com`. Suppose a message is sent from `me@email.com` to `mlist@email.com`. Then, the mailing list address `mlist@email.com` is expanded to {`them@email.com`, `him@email.com`, `her@email.com`}. In addition, the message frequency count is increased by one for each communication pair: {`me@email.com`, `them@email.com`}, {`me@email.com`, `him@email.com`}, {`me@email.com`, `her@email.com`} and {`me@email.com`, `mlist@email.com`}.

Figure 2 presents the general procedure for extracting structured information. First, the sender's address is extracted from the `From` and `Reply-To` fields (multiple email addresses for a user are mapped to the same user). Next, receiver addresses are extracted from all address fields. A receiver address that is a mailing list address is expanded to its corresponding membership addresses, and all the membership addresses are added to the receiver list (duplicate entries are discarded). Finally, for

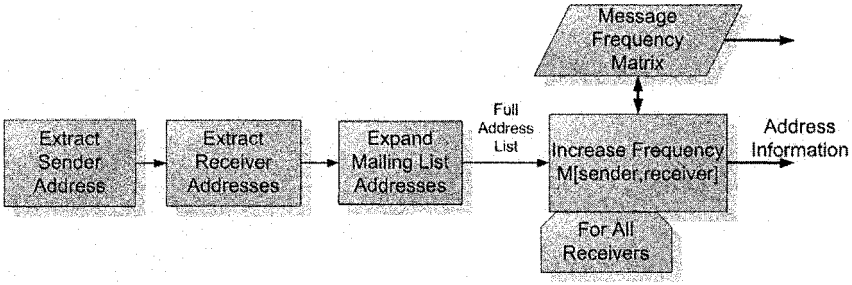


Figure 2. Structured information extraction.

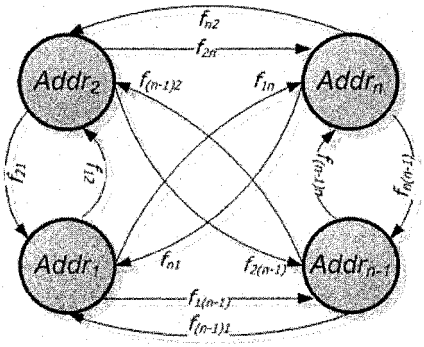


Figure 3. Social network graph from message frequency matrix.

each [`sender`, `receiver`] pair, the message frequency matrix is updated by incrementing the frequency value of the communication pair by one. After all the email messages are processed, the frequency values in the matrix are normalized based on the total number of communicating entities.

A directed network graph (social network graph) can be constructed from the set of communication pairs in the matrix (see Figure 3). Modeling address information as a directed social network graph helps establish asymmetric relationships between communicating parties (e.g., group leader and members). For example, a high message frequency between `me@company.com` and `you@company.com` may indicate that a strong relationship exists between the individuals. Since both email addresses belong to the same domain (`company.com`), one might infer that the two individuals are employees of the same company. It is normal to see high frequencies for emails between co-workers.

On the other hand, if `you@company.com` has a high frequency of communication with `him@competitor.com`, it could be inferred that `you@company.com` is passing secrets to `him@competitor.com` at a competing company.

Table 2. Example feature vector.

t	F[t]
$term_1$	$freq_1$
$term_2$	$freq_2$
$term_3$	$freq_3$

5.2 Unstructured Information Extraction

The unstructured sections in an email message include the `Subject` field, the message body and attachments. This paper focuses on plain-text content; the analysis of email attachments is a topic for future research.

The subject line in an email header can be considered to be unstructured information. The combination of the subject and message body is defined as the "feature string." The feature string contains terms that describe its content. These features (key terms) range from unique non-dictionary terms and numerical sequences to complex components such as web addresses (e.g., `www.google.com`). Relevant features must be extracted from the feature string to produce a summarized version of each message. Internet document indexing schemes (e.g., [1, 5]) can be applied to extract information from feature strings.

When processing a feature string, single terms should be extracted that best summarize the contents. This can range from using simple grammatical rules (e.g., ignoring articles and prepositions) to using a large dictionary list. When extracting these features, generalizations between words should be used. For example, the numerical sequence {`six`, `6`, `VI`} could be summarized as {`6`}, known nouns and acronyms {`U.S.`, `USA`} as {`USA`}, and verb variations {`run`, `running`, `ran`} as {`run`}.

Extracting semantic word pairs is essential to obtaining a full understanding of a feature string. For example, extracting the word pair {`Air Force`} is better than extracting {`Air`} and {`Force`} separately. This procedure can be implemented using natural language processing algorithms that recognize [`adjective`, `noun`] sequences, and provide table-lookups of known semantic word pairs.

After all the features are extracted from a feature string, a feature vector is constructed by extracting and recording the frequencies of each feature. Table 2 provides an example of a feature vector where $F[t]$ represents the frequency of feature t in the feature string.

The higher the frequency of a specific feature in a message, the greater the relevance between the topic of the email and that feature. A feature vector can be used as a comparison point when performing linkage anal-

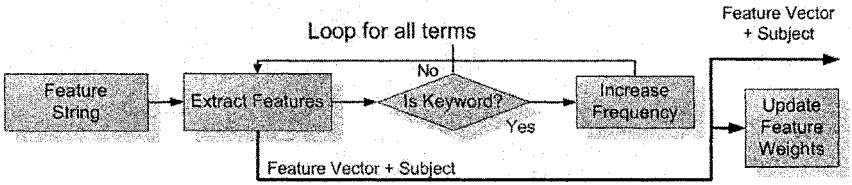


Figure 4. Unstructured information extraction.

ysis between email messages, i.e., clustering email message pairs that have strong relevance to each other.

Figure 4 shows the general procedure for extracting unstructured information. First, the subject and the message body are concatenated to produce a feature string, and the subject line is stored as the discussion thread in the feature vector. Next, each single term and semantic word pair in the feature string is checked to see if it can be considered to be a feature. Terms that are not considered keywords (or word pairs) are ignored. A term that is considered to be a keyword (or word pair) is extracted, and an entry is created for it in the feature vector for the email if none exists (and the frequency is set to one). If the feature already exists in the feature vector, the current frequency of that feature is incremented by one.

After all the features have been collected, the feature weights list is updated to include all the features extracted from the current email. The frequency count is increased by the frequency value in the feature vector for all the terms in the feature weights list.

5.3 Feature Weights

Suppose email evidence is collected from Company `XYZ`. Then, it is not appropriate to use the term `XYZ` as a unique feature. This is because `XYZ` has a high probability of being found in the evidence, e.g., because employees may mention the company's name in their correspondence or a disclaimer that mentions the company's name is appended to every email message. Therefore, the list of extracted features from an entire email set must be collected to produce feature weight statistics that help in evaluating feature uniqueness during link analysis.

Feature weights are calculated as follows. If k_i is a feature in feature set K, and f_i is the frequency of k_i. Then, the weight $\phi(k_i)$ of feature k_i is defined as $\phi(k_i) = 1 - \frac{f_i}{|K|}$.

6. Link Discovery

After email messages are processed for structured and unstructured data, various link analysis schemes can be used to discover hidden relationships between email users.

6.1 Link Analysis

Email evidence is analyzed by comparing each pair of email messages and calculating a suspicion value using the information contained in feature vectors and the message frequency matrix. This scheme potentially yields high true positive rates while maintaining acceptable false positive rates. These benefits come at the cost of performing $\binom{n}{2}$ comparisons.

6.2 Multiple Levels of Abstraction

Linkage analysis schemes often encounter obstacles when comparing email messages. Data specificity is a problem. For example, suppose one email has the feature `Merlot` and another has the feature `Chardonnay`. Comparing these features directly does not produce a correlation because the two features do not have direct lexical similarity. One solution is to improve the correlation between the feature vectors using multiple levels of abstraction (MLA).

Individuals write text in different contexts and perspectives; therefore, perfectly matching the features of two email messages will be very uncommon. Creating a decision tree based on taxonomic data can help produce higher correlation values between emails.

Several algorithms, e.g., the Attribute Value Taxonomy-Guided Decision Tree Algorithm [12], operate at different levels of specificity. WordNet[7], a lexical database for the English language, produces synonyms for various terms that represent a single lexical concept. These resources can be adapted to work with email messages at various levels of abstraction.

Consider the simple abstraction tree shown in Figure 5. It can be determined that `Merlot` and `Chardonnay` are both types of `Wines` using one level of abstraction ($\alpha = 1$). Therefore, if the features `Merlot` and `Chardonnay` are generalized to `Wines`, there is a direct lexical similarity, which produces a higher correlation match between the features. The increase in correlation can be fine-tuned using different levels of detail and precision in an abstraction tree. The main reason for using MLA is to reduce the inter-cluster distance between email messages by enhancing their relevance.

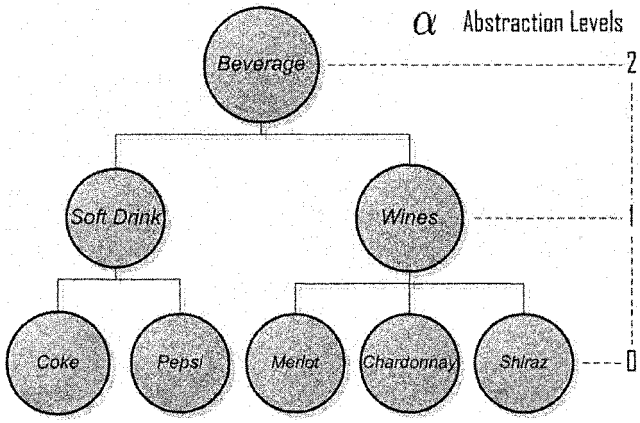


Figure 5. Example abstraction tree.

6.3 Suspicion Level

A suspicion level S, where $0 \leq S \leq 1$, is used to determine whether or not email pairs are correlated. The inter-cluster distance between message pairs is determined by comparing the set of features between feature vector pairs. The suspicion level is calculated by combining MLA with the corresponding weights of the intersection of related features.

Let F_i and F_j be feature vectors from emails i and j, where $F_x[y]$ is the frequency of feature y in F_x. Furthermore, define $A_{ij} = F_i \cup F_j$ and let $M_{ij} = F_i \cap F_j$ using MLA. Therefore, $M_{ij} \subseteq A_{ij} \subseteq K$, where K is the set of features extracted from the emails. Additionally, let $\phi(k_i)$ be the feature weight of feature k_i. Then, the suspicion level S is given by:

$$S = 0 \leq \frac{\sum_{p=1}^{|M_{ij}|} \phi(m_p) * \max(F_i[m_p], F_j[m_p])}{\sum_{q=1}^{|A_{ij}|} \phi(a_q) * \max(F_i[a_q], F_j[a_q])} \leq 1 \tag{1}$$

When performing link analysis, a threshold should be chosen to determine the suspicion level that is needed to consider an email message pair correlated.

Figure 6 outlines the procedure for performing link analysis for email messages. For each pair of email feature vectors (e_1, e_2), the subject and initial header information are used to determine whether or not the emails belong to the same discussion thread. If they are, the two emails are reported as correlated. If not, matching features are identified using the feature vectors for each message and MLA. Next, a suspicion level is computed using the message frequency matrix and the statistical feature weights of the matching features of the two email messages. If the

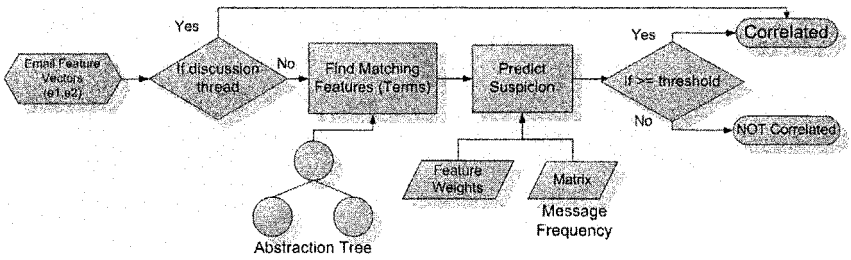


Figure 6. Link analysis.

Table 3. Results for different generalization heights.

α	**False Positives (4704)**	**True Positives (246)**	**PPV**
0	46 (0.98%)	74 (30.08%)	61.66%
1	106 (2.25%)	96 (39.02%)	46.60%
2	233 (4.95%)	143 (58.13%)	38.03%

suspicion value is higher than a user-defined threshold, the two emails are reported as correlated. Otherwise, the emails are reported as not correlated.

7. Results and Discussion

A training data set of 100 email messages was obtained from a personal inbox and modified manually for use in the experiment. A database of 150,843 words [4] was used to determine if a word was considered to be a feature in a feature string. A static two-level abstraction tree that specifically related to the content of the training email messages was implemented. A minimum threshold value of 0.7 (70% suspicion) was used as the cut-off point to decide whether or not two emails were correlated. Email pairs in different clusters reported as correlated were considered to be false positives. Email pairs in the same cluster reported as not correlated were considered to be false negatives. Pairs in the same cluster reported as correlated were designated as true positives.

The results obtained for various generalization heights (α) are presented in Table 3. Note that the positive predictive value (PPV) is defined as the probability that an email message pair is a true positive when restricted to the entire set of email message pairs reported as correlated. The main findings are discussed below.

A trade-off exists when using MLA. Using an abstraction tree in link analysis provides the ability to increase the total number of true positives at the cost of increasing the total number of false positives.

The higher the level of abstraction used in link analysis on the training set, the greater the number of email pairs that become relevant to each other. This increases the number of correct relationships found between email messages; however, more pairs of unrelated emails are identified as being relevant. For the training data set, an abstraction level (α) of one produced acceptable false positive and true positive rates.

The number of features extracted from email messages significantly affects correlation outcomes. It was found that the number of extracted features from an email increases the probability that an email message pair is correlated. The process of selecting features from a feature string is a user-defined process. If the selection process is not strict (i.e., most terms are considered features), then more features are generalized using MLA to increase correlations between message pairs.

8. Conclusions

Email messages contain valuable information that can be used to deduce social networks, and correlate events and activities in cyber crime investigations. The proposed email investigation framework, based on information extraction, linkage analysis, message frequencies and multiple abstraction levels, can automate the analysis of email evidence. While the initial results on email correlation are promising, statistical techniques must be employed to enhance linkage analysis based on abstraction trees and feature weights. Research efforts should also focus on integrating advanced text mining and data mining techniques in the email investigation framework.

Acknowledgements

This research was supported in part by NSF Grant DUE-0313837 and the GEM Fellowship Program. The authors also wish to thank Dr. Diane Cook and anonymous reviewers for valuable discussions and helpful comments on earlier versions of this paper.

References

[1] M. Berry (Ed.), *Survey of Text Mining: Clustering, Classification and Retrieval*, Springer-Verlag, New York, 2003.

[2] D. Boyd and J. Potter, Social network fragments (smg.media.mit.edu/projects/SocialNetworkFragments), 2004.

[3] Cable News Network, Merck's Vioxx e-mail scrutinized (money.cnn.com/2004/11/01/news/fortune500/merck), November 1, 2004.

[4] CTAN, English dictionary list (ctan.tug.org/tex-archive/systems /win32/winedt/dict/english.zip).

[5] B. Davison, Unifying text and link analysis, *Proceedings of the IJCAI Workshop on Text Mining and Link Analysis* (www.cs.cmu.edu/~dunja/TextLink2003/Papers/DavisonTextLink03.pdf), 2003.

[6] Federal Energy Regulatory Commission, Information released in the Enron investigation (www.ferc.gov/industries/electric/indus-act /wec/enron/info-release.asp), 2005.

[7] C. Fellbaum (Ed.), *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge, Massachusetts, 1998.

[8] V. Krebs, Discovering social networks and communities in email flows (www.orgnet.com/email.html), 2003.

[9] J. Postel, Simple Mail Transfer Protocol, RFC 821, August 1982.

[10] P. Resnick, Internet message format, RFC 2822, April 2001.

[11] Wall Street Journal, Merck down 6.4% on report company knew of Vioxx risk early (online.wsj.com), November 2004.

[12] J. Zhang and V. Honavar, Learning from attribute value taxonomies and partially specified instances, *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 880-887, 2003.

Chapter 8

THE MITNICK CASE: HOW BAYES COULD HAVE HELPED

Thomas Duval, Bernard Jouga and Laurent Roger

Abstract Digital forensics seeks to explain how an attack occurred and who perpetrated the attack. The process relies primarily on the investigator's knowledge, skill and experience, and is not easily automated. This paper uses Bayesian networks to model the investigative process, with the goal of automating forensic investigations. The methodology engages digital evidence acquired from compromised systems, knowledge about their configurations and vulnerabilities, and the results of previous investigations. All this information is stored in a database that provides a context for an investigation. The utility of the methodology is illustrated by applying it to the well-known Kevin Mitnick case.

Keywords: Computer crime investigations, Bayesian networks

1. Introduction

Two important goals in digital forensic investigations are to explain definitively how a computer system or network was attacked and to identify the perpetrators [9, 15]. The investigative process has certain subjective elements, which draw on the investigator's knowledge, skill and experience [8], and are not easily automated.

This paper uses Bayesian networks to model the investigative process, with the goal of automating forensic investigations. The XMeta system described in this paper engages digital evidence from compromised systems, knowledge about their configurations and vulnerabilities, and the results of previous investigations. Given the facts of a case, XMeta reasons about the situation, providing information about likely attacks, additional actions performed by attackers, the most vulnerable software systems, and the investigation techniques that should be used.

The following sections discuss Bayesian networks, and the XMeta model and implementation. The methodology is illustrated by applying it to the well-known Kevin Mitnick case [12, 13].

2. Bayesian Networks

Bayesian networks are directed acyclic graphs whose nodes are variables and links are causal connections weighted with conditional probabilities [2]. Bayesian networks are useful for modeling complex situations for which information is incomplete and/or uncertain.

For example, suppose a web page on an Apache server A has been defaced. Assume that there are two possible causes: (i) the attacker used an exploit E, or (ii) the attacker stole the administrator's password S. A well-formed Bayesian network provides a probabilistic answer. Specifically, according to Bayes' Theorem, the probability that an exploit E is performed given the presence of web server A is:

$$P(E \mid A) = \frac{P(E, A)}{P(E)}$$

where $P(E, A)$ is the probability of having exploit E and web server A, and $P(E)$ is the probability of having exploit E.

The construction of a Bayesian network involves three steps: (i) constructing a causal graph, (ii) constructing probability tables associated with nodes, and (iii) propagating probabilities. A Bayesian network is typically constructed by interviewing domain experts to obtain information about nodes, causality links and probability values. Alternatively, the network structure and probabilities may be learned from examples, e.g., from a cases database.

An example Bayesian network is presented in Figure 1. In this network, if a `DoS` attack has occurred in addition to an `Apache` server compromise and a `web defacement`, the probability values of the `Exploit` and `Usurp` nodes will change accordingly. Also, the probability values of the software nodes (e.g., `MS Office`) will change to reflect their vulnerability to the `DoS` attack. Thus, inferences in a Bayesian network proceed in the top-down and bottom-up directions.

Bayesian networks have been already widely used in expert systems, including several security and forensics applications [1, 3, 14]. Costa and co-workers [5] have used Bayesian networks to reason about communications between hosts. Our work deals with communications between systems as well as system events.

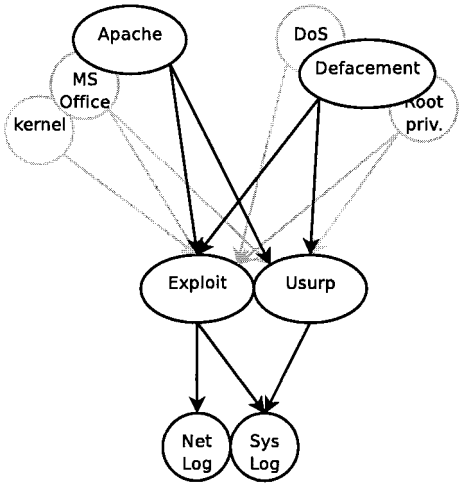


Figure 1. Bayesian network.

3. XMeta Model

XMeta uses a Bayesian network to model and reason about computer systems and networks that are compromised by attacks. A system compromised by a particular attack is modeled using an investigation plan. An investigation plan is a Bayesian network built on demand at the start of system analysis, which takes into account the system configuration.

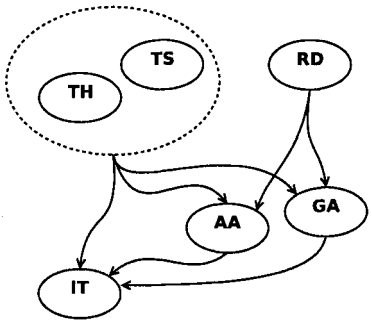


Figure 2. Investigation plan.

Figure 2 shows the structure of an investigation plan. It has six types of nodes: targeted hardware (TH), targeted software (TS), reported damage (RD), generic attacks (GA), additional actions (AA), and investigation techniques (IT).

Table 1. Attack nodes.

Variable	Description
listing	List a DNS entry (example)
net_listen	Listen on the network to get a password (example)
decrypt	Use a dictionary or a brute force attack to obtain passwords
exploit	Use an exploit to enter a system (e.g., buffer overflow)
bypass	Bypass a security element
broadcast	Broadcast packets (e.g., ping)
chaff	Use a fake server to steal information
embezzlement	Man-in-the-middle (example)
listen	Listen for host events
parasite	Transform software functionality
degrade	Alter a network/host-based service (e.g., web defacement)
diversion	Use a diversion
intercept	Intercept data
usurp	Use someone else's identity
bounce	Log into multiple hosts before attacking
trojan	Use a Trojan horse to install software
repeat	Scanning sweeping (example)
blocking	Block a network service
overrun	DoS, DDoS (examples)
brute_force	Use a brute force attack
control	Intercept and block a host

Table 2. Action nodes.

Variable	Description
msg	Send a message that signifies an attack
attribute	Escalate privileges
scan_use	Find host services by scanning a host
encrypt	Encrypt data
hidden_channel	Use a protocol weakness to send data
infection	Add information in a file (e.g., steganography)
illic_cnx	Connect to a host without approval
trap	Use a trap door
invert_trap	Use an inverted trap door
inhib_detect	Inhibit detection (e.g., IP spoofing)
del	Delete data
login_inst	Install a new login

A typical investigation plan has 40 to 50 nodes. The French Ministry of Defense (DGA) has identified 21 possible generic attacks (Table 1). Also, DGA has listed a set of 12 additional actions (Table 2).

Table 3. Investigation techniques.

Variable	Description
`image`	Make a forensic copy of media
`syst_check`	Check system log files
`net_check`	Check network log files (e.g., firewall logs)
`syst_var`	Check system variables (e.g., logins, processes)
`retrieve`	Retrieve hidden or deleted files
`net_log`	Use a sniffer to listen to attackers' actions
`int_topo`	Check the compromised network topology
`ext_topo`	Check the compromised network interconnections
`comm`	Analyze communications (e.g., IRC, mail logs)
`physic`	Analyze physical access to the computer

A new investigation plan is created by entering the host configuration (TH and TS) and the observed damage (RD). Much of this information can be obtained from the ICAT vulnerability database [11], which contains data about more than 7,000 software systems. A database of previous cases is used to set causality links and probability values (via the K2 learning algorithm [4, 10]). The Bayesian network uses the likelihood weighting approximate inference technique [6] to reason about attacks (GA) in Table 1 and actions (AA) in Table 2. Attacks (GA) are mandatory to compromise a host. On the other hand, actions (AA) are not mandatory, although their presence can assist investigations (e.g., an "I 0wn y0u!" message was sent, or a new login was created). Based on the data provided, XMeta proposes the investigation techniques (IT) that may be used. The list of investigation techniques is presented in Table 3.

When an investigator checks a particular attack or action, this fact is entered in the system, which correspondingly adjusts the values in the Bayesian network. When a host has been checked completely, i.e., the source of the attack has been identified, the following logical process is followed.

- If the source attack address is local, and
  - If the attacker had legitimate access, then the investigation is complete.
  - If the attacker gained access before launching the attack, then the investigation must continue and a new investigation plan is created using the same software configuration.

- If the source attack address is not local (i.e., internal or external), then the next step in the investigation depends on whether or not the next host is accessible for investigation.

Investigators can create as many investigation plans as needed. These plans may be linked to reflect the attack progression (i.e., multiple links are allowed).

4. XMeta Testbed

An XMeta testbed was developed using a Bayesian toolkit [7] for inference and a Python/GTK-based GUI. A newer version of the XMeta testbed, currently under development, is only based on Python/GTK.

Ideally, a Bayesian inference system should be initialized using the results of previous investigations. However, in the absence of real data, the ICAT vulnerability database [11] was used. The database provided information about software vulnerabilities and losses, as well as attacks and actions. Next, the K2 learning algorithm [4, 10] was used to set causality links and probability values. The only facts that could not be extracted from ICAT pertained to investigation techniques.

The initial version of the XMeta testbed only considered the most vulnerable software systems (based on the number of vulnerabilities and observed losses), and possible attacks and actions of attackers. In the following, a fictitious case is used to demonstrate the types of results obtained with the initial testbed.

Consider a situation where confidential information was stolen from a workstation. The compromised host ran a Linux Debian (without patches). The investigation plan was initialized with Debian software (kernel, libc, Windowmaker, OpenSSH) and a confidentiality loss. The XMeta system indicated that the source of the attack was probably local to the machine, which was true. XMeta also indicated the most vulnerable software systems were XFree86, Linux libc and the kernel (the kernel was actually compromised). Finally, XMeta identified the actual attack (the attacker used an exploit to become root and copy the stolen file) as the seventh most likely of the 21 possible attacks (see Table 1). Clearly, all 21 attacks should checked in detail in a real investigation. Depending on the context, some attacks can be eliminated, e.g, a DoS attack is not relevant to a data theft investigation.

Since the initial implementation, the XMeta database has been augmented significantly, including adding information about investigation techniques. The results of the current XMeta testbed are much better than those of the initial version.

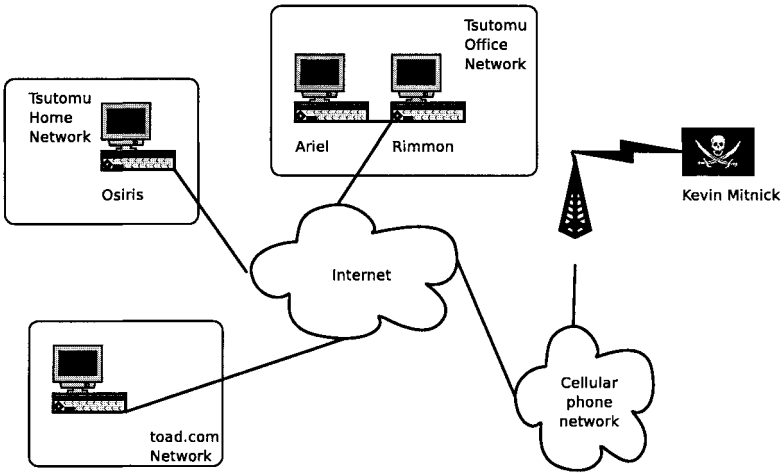


Figure 3. Computer system involved in the Kevin Mitnick case.

5. The Kevin Mitnick Case

In 1994, an unknown attacker hacked into computers at the San Diego Supercomputer Center. After seven weeks of intensive investigations, Tsutomu Shimomura, who worked at the center, tracked the perpetrator, Kevin Mitnick, to an apartment in Raleigh, North Carolina. Mitnick was arrested on February, 14 1995. He was convicted of breaking into some of the United States' most secure computer systems [12, 13].

This section describes the application of XMeta to the Mitnick case. The case is interesting because of its complexity and the number of systems involved. It provides an excellent context for comparing XMeta's results with those from Shimomura's original investigation.

5.1 The Mitnick Investigation

Upon examining compromised systems at the San Diego Supercomputer Center, Shimomura noticed that numerous network scans had been conducted the previous night. One of the first clues was found on a computer named `Ariel`. A large file (`oki.tar.Z`) had been created. This file was transferred to an unknown address, and then deleted from `Ariel`. It was later discovered that `oki.tar.Z` contained confidential data about cell phone firmware.

The following information pertaining to the Mitnick investigation was provided to XMeta. Note that the information was obtained from public sources [12, 13], and is incomplete and/or inexact.

Software: SunOS, GNU tar, GNU ghostscript, fingerd, ruserd, ftp
Losses: LT_Confidentiality

The following results were provided by XMeta:

Ariel:
The probable **attacks** are: `bypass` (65%), `diversion` (56%), `brute_force` (56%).
The probable **additional actions** are: `infection` (83%), `inhib_detect` (81%), `login_inst` (71%).
The highlighted **software systems** are: GNU tar (73%), finger service (73%), ftp (27%).
The proposed **investigation techniques** are: none.

Note that investigation techniques were not proposed by XMeta because a similar case did not exist in its database at the time.

Xmeta's answers indicate that the log files should be checked for suspicious applications. Based on the three attacks that are listed, one might infer that the attack came from outside (this assumption is strengthened by the network scans that were observed). The attacker probably bypassed the security between `Ariel` and another computer, or made a diversion to upload the data file. The brute force attack can be dismissed because it is not possible to enter a system using such an attack. Note that XMeta indicated a brute force attack was possible because its database was populated mainly with ICAT data; for the specified software systems and loss, a brute force attack is one of the three most common attacks in ICAT.

```
14:09:32 toad.com# finger -l @target
14:10:21 toad.com# finger -l @server
14:10:50 toad.com# finger -l root@server
14:11:07 toad.com# finger -l @x-terminal
14:11:38 toad.com# showmount -e x-terminal
14:11:49 toad.com# rpcinfo -p x-terminal
14:12:05 toad.com# finger -l root@x-terminal
```

Figure 4. `toad.com` logs.

Shimomura observed that the network scans originated from a host in domain `toad.com` (see Figure 4 [12]). In Figure 4, target refers to `Ariel`, server to `Rimmon`, and x-terminal to `Osiris`. Shimomura also observed that his computer (`Osiris`) exhibited strange behavior – blank windows on the top of the screen. The facts are:

- `Ariel` and `Osiris` had strong relationships
- `Osiris` was located at Shimomura's home and had no direct connection with `toad.com`

```
14:18:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S 1382726990:1382726990(0)
        win 4096

14:18:26.094731 x-terminal.shell > apollo.it.luc.edu.1000: S 2021824000:2021824000(0)
        ack 1382726991 win 4096

14:18:26.172394 apollo.it.luc.edu.1000 > x-terminal.shell: R 1382726991:1382726991(0)
        win 0

14:18:26.507560 apollo.it.luc.edu.999 > x-terminal.shell: S 1382726991:1382726991(0)
        win 4096

14:18:26.694691 x-terminal.shell > apollo.it.luc.edu.999: S 2021952000:2021952000(0)
        ack 1382726992 win 4096

14:18:26.775037 apollo.it.luc.edu.999 > x-terminal.shell: R 1382726992:1382726992(0)
        win 0
```

Figure 5. `Osiris` logs.

- Considerable network traffic was directed at `Osiris` (Figure 5).

These facts imply that the investigation should continue with `Osiris` and not (for the moment) with `toad.com`. Furthermore, `Osiris` may be the source of the attack or an intermediate system in the attack.

Consequently, a new investigation plan is created for `Osiris`. In fact, Shimomura discovered that `Osiris` was disconnected from his office network and especially from `Ariel`.

The following facts were provided to XMeta:

> **Osiris**:
> **Software**: SunOS, GNU tar, GNU ghostscript, fingerd, ruserd, ftp
> **Losses**: LT_Availability

The following results were provided by XMeta:

> The probable **attacks** are: `repeat` (e.g., scanning sweeping) 100%, `overrun` (e.g., DoS, DDoS, smurf, fraggle) 89%, `bypass` (68%).
> The probable **additional actions** are: `infection` (73%), `trap` (backdoor) 62%, `del` (data deletion) (45%).
> The highlighted **software systems** are: ftp (73%), GNU tar (38%), GNU ghostscript (38%).

The results indicate that `Osiris` was an intermediate system because an attacker cannot penetrate a host using scanning sweeping or overrun. Therefore, it is necessary to search for another computer.

`Osiris` was a X-Window terminal connected to `Rimmon`; possibly, it was also attacked. This is confirmed by Shimomura's logs (Figure 6).

Since information is not available about `Rimmon`, it is reasonable to assume that it had the same configuration as `Osiris` and `Ariel`. Shimomura discovered that an unauthorized user succeeded in installing a kernel module named `Tap 2.01` on `Rimmon` (Figure 7). This implies that the unauthorized user had root privileges.

```
14:18:22.516699 130.92.6.97.600 > server.login: S 1382726960:1382726960(0) win 4096
14:18:22.566069 130.92.6.97.601 > server.login: S 1382726961:1382726961(0) win 4096
14:18:22.744477 130.92.6.97.602 > server.login: S 1382726962:1382726962(0) win 4096
14:18:22.830111 130.92.6.97.603 > server.login: S 1382726963:1382726963(0) win 4096
14:18:22.886128 130.92.6.97.604 > server.login: S 1382726964:1382726964(0) win 4096
```

Figure 6. Rimmon logs.

```
x-terminal% modstat
Id  Type  Loadaddr      Size   B-major  C-major  Sysnum   Mod Name
 1  Pdrv  ff050000      1000              59.             tap/tap-2.01 alpha

x-terminal% ls -l /dev/tap
crwxrwxrwx  1 root      37,  59 Dec 25 14:40 /dev/tap
```

Figure 7. Rimmon system variables.

The following facts were provided to XMeta:

Rimmon:
Software: SunOS, GNU tar, GNU ghostscript, fingerd, ruserd, ftp
Losses: LT_Obtain_all_priv, LT_Availability

The following results were provided by XMeta:

The probable **attacks** are: `trojan` (93%), `bypass` (78%), `brute_force` (58%).
The probable **additional actions** are: `login_inst` (58%), `infection` (51%) and `trap` (46%).
The highlighted **software systems** are: ftp (59%), GNU tar (41%).

From these results, one can infer that if a Trojan horse was not found in `Rimmon`, the computer was used as an intermediate system like `Osiris`. Since Shimomura did not find a Trojan horse but a flooding attack (known as `overrun` in XMeta), it appears that `Rimmon` was used as an intermediate system to gain access to `Osiris` and `Ariel`. In fact, XMeta indicated that the overrun attack was the tenth most likely of the 21 possible attacks. (According to XMeta, the ten most likely attacks were: `trojan`, `bypass`, `brute_force`, `broadcast`, `chaff`, `repeat`, `intercept`, `net_listen`, `bounce` and `overrun`. However, Shimomura also discovered that the attacker had installed a kernel module in `Rimmon` and, therefore, had root access.

5.2 XMeta's Results

XMeta discovered the following elements in the Mitnick case.

- A file `oki.tar.Z` was transferred from `Ariel` to an unknown address using a bypass attack or a diversion attack.
- A host in the `toad.com` domain was used to scan the network.

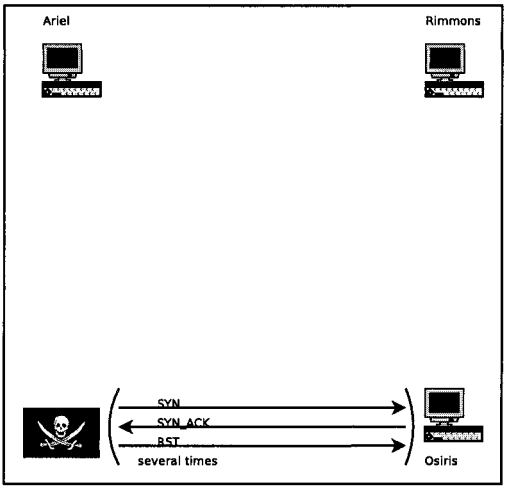


Figure 8. Mitnick attack (first step).

- The attacker either used the repeat attack on `Osiris` to obtain information or bypassed security to enter `Osiris`.
- The attacker exploited the trust relationship between `Osiris` and `Rimmon` to access `Osiris`.
- The attacker installed a kernel module and used it to access `Ariel`.

These elements can be saved, giving future users of the system the ability to replay the attack or the entire investigation (e.g., for a trial). To support this goal, we have defined the Computer Forensics XML Report (CFXR) System, which uses an XML-based format to save system configurations, attacks, additional actions, investigation techniques, as well as the progressions of attacks and investigations.

The next steps in the Mitnick investigation are to determine how the attacker gained root access to `Osiris` and the `toad.com` host, and the destination of the `oki.tar.Z` file.

5.3 Shimomura's Results

Shimomura [12, 13] broke down the attack into three steps. In the first step, the attacker tried to guess the initial TCP sequence numbers for incoming connections to `Osiris`. This was accomplished by sending SYN packets followed by a connection reset (Figure 8).

In the second step, the attacker spoofed `Rimmon` to open a connection to `Osiris`. Next, the attacker used `rsh` to issue the command `echo ++ >>/.rhosts` and gain root privileges. This attack (bypass)

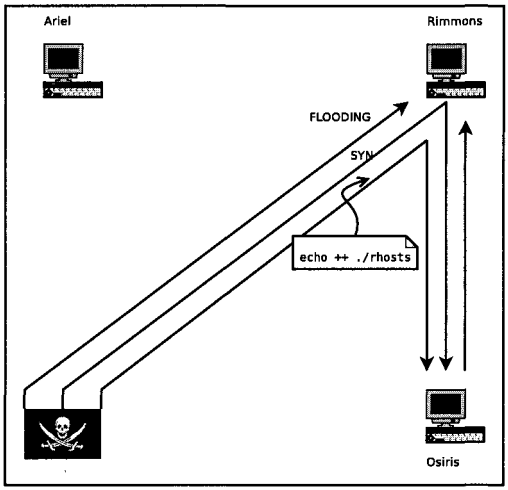


Figure 9. Mitnick attack (second step).

was identified by XMeta as the third most likely attack (68%). Flooding (`overrun` in XMeta) was used to gag `Rimmon` during the three-way handshake when establishing the TCP connection with `Osiris` (Figure 9).

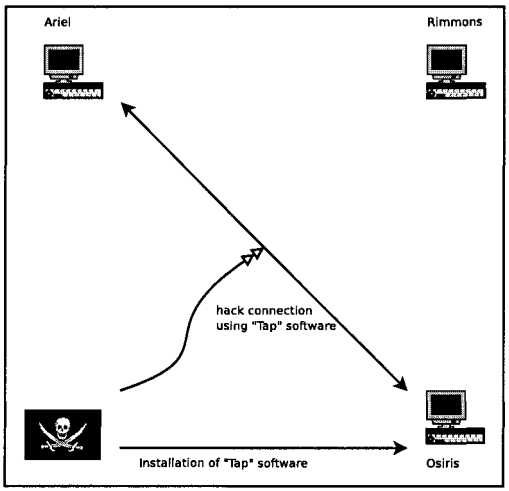


Figure 10. Mitnick attack (third step).

In the third step, the attacker installed `Tap` software and used it to hack the connection between `Osiris` and `Ariel` (Figure 10). The attacker thus gained access to `Ariel`, and created and downloaded the `oki.tar.Z` file.

6. Conclusions

The XMeta system uses a Bayesian network to reason about attacks on computer systems and networks. In particular, it provides information about likely attacks, additional actions performed by attackers, the most vulnerable software systems, and the investigation techniques that should be used. The application of XMeta to the investigation of the Kevin Mitnick case demonstrates its utility as a digital forensic expert system. A supporting Computer Forensics XML Report (CFXR) System uses an XML-based format to save system configurations, attacks, additional actions, investigation techniques, and the progressions of attacks and investigations.

Additional research is needed to enable XMeta to support forensic investigations. The database of cases must be enhanced to obtain better results, especially the suggested investigation techniques. It is also necessary to incorporate criminal profiles that will help refine the assumptions, resulting in more accurate information about targets and attacks.

References

[1] D. Burroughs, L. Wilson and G. Cybenko, Analysis of distributed intrusion detection systems using Bayesian methods, *Proceedings of the Twenty-First IEEE International Performance, Computing and Communications Conference*, 2002.

[2] E. Charniak, Bayesian networks without tears, *AI Magazine*, vol. 12(4), pp. 50-63, 1991.

[3] A. Christie, The Incident Detection, Analysis and Response (IDAR) Project, Technical Report, CERT Coordination Center, Carnegie Mellon University, Pittsburgh, Pensylvania, 2002.

[4] G. Cooper and E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, *Machine Learning*, vol. 9(4), pp. 309-347, 1992.

[5] P. Costa, J. Jones, B. Liao and V. Malgari, A system for collection, storage and analysis of multi-platform computer system data, Technical Report, George Mason University, Fairfax, Virginia, 2003.

[6] R. Fung and K. Chang, Weighing and integrating evidence for stochastic simulation in Bayesian networks, *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 209-219, 1989.

[7] W. Hsu, Bayesian Network Tools in Java (bndev.sourceforge.net).

[8] T. Levitt and K. Laskey, Computational inference for evidential reasoning in support of judicial proof, *Cardozo Law Review*, vol. 22(5), pp. 1691-1732, 2001.

[9] K. Mandia and C. Prosise, *Incident Response: Investigating Computer Crime*, McGraw-Hill/Osborne, Emeryville, California, 2001.

[10] P. Naim, P. Wuillemin, P. Leray, O. Pourret and A. Becker, *Reseaux Bayesiens*, Eyrolles, Paris, France, 2004.

[11] NIST, National Vulnerability (formerly ICAT) Database (nvd.nist.gov).

[12] T. Shimomura, Technical details of the attack described by Markoff in NYT (blinky-lights.org/shimomura-25jan95.html), 1995.

[13] T. Shimomura and J. Markov, *Takedown*, Hyperion Press, New York, 1996.

[14] SpamAssassin, The Apache SpamAssassin Project (spamassassin.apache.org).

[15] U.S. Department of Justice, Computer Crime and Intellectual Property Section (www.cybercrime.gov).

Chapter 9

APPLYING FORENSIC PRINCIPLES TO COMPUTER-BASED ASSESSMENT

R. Laubscher, D. Rabe, M. Olivier, J. Eloff and H. Venter

Abstract A computer forensic investigator investigates computer crime. Currently only a few academic institutions have a computer forensic department and, therefore, the investigative responsibility (in case of contravention of assessment regulations for computer-based assessments) rests upon the lecturers.

The purpose of our project is to apply computer forensic principles to a computer-based assessment environment to facilitate the identification and prosecution of any party that contravenes assessment regulations. This paper is the first step in that project; its purpose is to consider the nature of such an environment. This nature is derived from the established computer forensic principles. In particular, we focus on the forensic process to determine the policies, procedures and types of tools that should be present in such an environment. The intention of the paper is not to consider any of the issues raised in detail, but to consider the process from a high level. The utilization of different tools, namely a key logger, CCTV camera, audit log and a report of logins, facilitates the identification of any party that contravenes assessment regulations. The proposed process consists of four phases: preparation of the environment, collection of evidence, analysis of evidence, and reporting findings.

Keywords: Computer-based programming assessment, forensic process, key logging

1. Introduction

In learning to program, the greatest amount of time is devoted to working on the computer. As Wyer and Eisenbach [11] pointed out, it is more appropriate to test programming skills by means of a computerized assessment than by means of a paper-based assessment. Examples of risks that could arise in a computer-based assessment are: hidden pre-written code with the aim to copy or access the code during the

assessment, electronic communication with the aim of exchanging solutions or tips, impersonation of another learner (knowing the other user's login name and password) and presenting the programming project of another learner under this learner's name (masquerading). Even with security measures in place (e.g., the presence of invigilators, access control and authentication), the risk of contravention of assessment regulations is high, because learners find innovative methods to violate assessment regulations or bypass security controls.

Currently only a few academic institutions have a computer forensic department that is able to investigate suspected assessment misconduct. Computer forensic departments are emerging in the financial sector, but are likely to remain uncommon in academic institutions. Therefore the responsibility of conducting a computer forensic investigation will rest on the lecturer – in particular, the collection and analysis of the required computer evidence. After an investigation is completed, the investigation report could be presented to the examination board and authorities, and may lead to a court case.

One of the academic responsibilities of a lecturer is to certify that each learner has mastered the subject area to the degree reflected in the marks awarded. An assessment invigilator (or commissioner) has a dual duty to fulfill. On the one hand the invigilator must provide an environment in which the learner can be treated with his/her right to privacy during the assessment to enable the candidate to complete the assessment with as few distractions as possible. On the other hand, the invigilator must also be able to determine, beyond reasonable doubt, which resources, legitimate and illicit, were used to develop a programming project.

The purpose of our project is to apply forensic principles to a computerized assessment environment in order to facilitate the identification and prosecution of any party that contravenes assessment regulations. This paper is the first step in that project; its purpose is to consider the constituent elements of such an environment. These elements are derived from established computer forensic principles. In particular we will focus on the forensic process to determine the policies, procedures and types of tools that should be present in such an environment. The intention of the paper is not to consider in detail any of the issues raised, but to consider the process from a high level.

The remainder of this paper is structured as follows. Section 2 defines the relevant terminology. Section 3 describes the computer forensic process in a computerized assessment environment. Sections 4, 5, 6 and 7 elaborate on the four phases of the assessment forensic process. Section 8 presents the conclusions and recommendations.

2. Relevant Terminology

Computer Forensics: The goals of computer forensics are to conduct a structured investigation to determine exactly what happened and who was responsible, and to perform the investigation so that the results are useful in a criminal proceeding [6] (p. 292). In our case, the possible culprits are known and the crime domain is restricted.

In a forensically sound methodology, no changes are made on data from the original evidence; the evidence is preserved in a pristine condition [5]. It should also not matter who completes the examination of media, which tools are used and which methods employed – the same results should always be obtained.

Key Logging Tools: Key loggers record every keystroke and mouse action performed by the user of the computer on which these tools are activated. Examples of key logger software are: KeyCapture [9], Powered Keylogger, Handy Keylogger, Stealth Keylogger and Perfect Keylogger [8]. The emphasis is on the key logger as primary source for evidence collection and the other tools (CCTV camera and audit logs) as secondary sources.

Audit Logs: The general notion of an audit log is appealing for use in an assessment environment. In practice, however, an audit log may be difficult to handle, owing to the volume of data and analysis effort required. To overcome this problem, we suggest that a backup of the audit log is made and then cleared before the computer-based programming assessment commences. Only transactions within the specific computer-based programming assessment time-slot should be recorded for forensic investigation purposes.

3. Proposed Computer Forensic Process

For the purposes of this research the following prerequisites are assumed to establish a controlled computer-based programming assessment environment: there are at least two invigilators, of which the programming lecturer is one, learners complete the assessment on randomly assigned (by the lecturer) workstations that are not connected to the Internet, the assessment starts and ends at a specific time, network access is restricted to a designated folder for each learner (with appropriate access rights), learners are not able to observe any other screens and casting of a previously set-up computer is utilized in order to ensure identical configurations on each workstation.

According to Holley [5], a computer forensic investigation is a four-step process conducted after a crime has been reported and a subpoena issued. The first step is to identify which computer media may contain

evidence. The second step is to preserve the digital data because they are fragile. The third step is to analyze the data by conducting an examination either directly on the image or to use the image to make another copy of the media to be examined. The final step is to report the findings to decision-makers for action.

According to Armstrong [1], when forensics is used to its full potential, it can provide both pre- and post-event benefits. The computer forensics process we are proposing for computer-based programming assessment also consists of four phases: preparation of the environment, evidence collection, analysis, and reporting findings.

During the first phase the controlled environment is prepared prior to the assessment. The activities are casting all computers with a previously set-up computer, disabling Internet connections, activating key logger software, audit log and CCTV camera, verifying that the time and dates of all computers are correct and identical, announcing assessment regulations, and obtaining the consent of learners for evidence collection.

In the second phase the computer-based programming assessment starts. Key loggers capture the learner's keyboard and mouse actions and the electronic activities are recorded in the audit log. Login activities must be monitored frequently by generating a report indicating details of all users logged onto the network. For the duration of the assessment session, the CCTV camera records all activities within the computer laboratory.

The third phase in the forensic process starts after the completion of the assessment. Back-ups of all files (i.e., key logger files, audit log, login reports, programming projects of learners) are made on a separate computer or other trusted computer-storage media. Only then are the CCTV camera, key logger and audit log disabled. Next, an initial systematic scanning of all electronic evidence collected should be conducted and analyzed for suspected activities that transgresses regulations for the assessment. It is possible to confirm deviations found in one evidence source by cross-checking with the other sources of evidence. If dishonesty is suspected, a comprehensive forensic analysis must be conducted.

In the fourth phase, the findings are reported to the examination board and authorities.

The purpose of the proposed four-phase process is to collect different types of evidence for analysis, which act as indicators that regulations have been transgressed. Confirmation could be achieved by cross-checking the different sources of evidence. This forms the basis of proving that transgression of regulation has been committed in the computer-based programming assessment. The comprehensive forensic investigation lies beyond the scope of this paper. Now that an overview

of the process has been given, the following sections elaborate on each of the phases.

4. Phase I: Preparation for Evidence Collection

Preparation for evidence collection starts prior to the assessment. When a computer cast is done from a previously set-up computer, all workstations start with an identical image to prevent learners from hiding pre-written code or programs. Care must be taken to ensure that the computer from which the cast is created is virus free. The presence of viruses increases the complexity of the burden of proof. Next, a CCTV camera, audit log and key logger are activated for every workstation. The CCTV camera is used for surveillance (who sits where), the audit log records network transactions, and the key logger monitors the specific workstation.

Workstations must be assigned to learners upon their arrival. The learners should sign consent forms to permit the utilization of key logger tools to monitor all keyboard and mouse actions. An institutional policy should be in place and explained to the learners upon registering at the institution; they should also sign a document acknowledging that they understand the policy and will adhere to it.

Other important issues in the preparation phase are discussed in the following subsections: assessment regulations, legal issues, and time and date stamps.

Assessment Regulations: Examples of assessment regulations and instructions include a prohibition of communication between candidates in the assessment room as well as a prohibition on the use of supporting material (blank paper, books, notes, calculators, cellular phones and other electronic equipment) in the assessment room. Regulations specific to computer-based assessment usually explicitly prohibit electronic communication between candidates themselves and electronic communication between candidates and other external people. Electronic access to digital documents is restricted to those explicitly specified in the assessment paper.

Legal Issues: Obviously, invasion of privacy is a serious legal issue. To overcome the privacy issue, it is necessary to ensure that there are written institutional policies stating the rights of the institution or auditors to access and review all data on the institution's computers and related media and peripherals at any time, and to monitor all actions while using the facilities to ensure compliance with institutional policy [2].

Time and Date Stamps: Evidence collection relies extensively on time and date stamps of objects. Boyd and Forster [3] suggest that special care should be taken to ensure the authentication and integrity of the time and date stamps of the objects. Before the assessment starts, the CMOS time on each workstation and the server should be verified and synchronized in relation to actual time, obtainable using radio signal clocks or via the Internet using reliable time-servers. Learners should not have write access to the time settings.

5. Phase II: Evidence Collection

During the assessment, further monitoring activities may be applied to enhance evidence collection and analysis. Protection should be in place to ensure that learners could not bypass the key logging process, disable it or tamper with the log file. The captured keystrokes and mouse clicks should be written to a predetermined file on the server, protected by access control to ensure its authenticity and integrity. Masquerading will be confirmed if login attempts of more than one user are recorded for the same file.

Impersonation of learners during computer-based assessment could be identified if the lecturer frequently monitors login activity and writes the evidence to a separate file. Alternatively, the evidence will also be in the audit log. The advantage of monitoring logins during the session is that a learner could be caught in the act of impersonation.

6. Phase III: Analysis of Collected Evidence

After the submission of the final version of the programming project, five steps remain in the proposed computer forensic process: preserving computer media, de-activating logging devices, conducting an initial analysis of the collected evidence, conducting a comprehensive analysis of the evidence, and reporting findings.

Analysis should be conducted on exact copies of the media. Reliable backup copies should be created for investigations of key logger files, the audit log, the file containing the login reports, and the learners' final programming projects. These files should be inaccessible to the learners or any other person or object. A message digest (MD) could be calculated for the evidence data, providing a seal and encasing it so that any change is apparent [7] (p. 76).

It is necessary to verify that the final project submitted is the learner's legitimate and own work, created during the assessment. For this purpose it should be verified that non-permitted objects were not accessed or copied during the assessment, no electronic communications occurred

between the learner and another person, one learner did not impersonate another learner, and the learner did not employ other tools, e.g., program generators, during the session.

Two processes are involved in evidence analysis: (i) an initial search for possible dishonesty, seeking clues that indicate communication, copying or accessing non-permitted files, and (ii) a comprehensive analysis when there is a possibility of dishonesty.

The initial analysis strategy involves replaying the learner's key strokes and mouse clicks. This is performed on a workstation that is in a "clean state" (configured similar to workstations at the beginning of the assessment session). Replaying the key strokes and mouse clicks should therefore result in the same file(s) that were submitted by the candidate. Further analysis is required of any differences are found between the submitted file(s) and those created during the replay.

The initial analysis could be concluded by scanning the audit log for all disk accesses and broadcast messages, and by viewing the last entry in the file which reports the logins to confirm that all learners have logged in only once at the beginning of the assessment.

The final phase in the forensic process is to report the findings to the decision-makers, i.e., the examination board and authorities. The reporting of findings, discussed in the following section, is crucial to the forensic process.

7. Phase IV: Reporting Findings

All the forensic activities and the evidence collected should be documented with precision and accuracy. The report must be written in a clear and concise manner and should be understandable to non-technical people.

Of course, underlying all of these activities is "chain-of-custody." This refers to the proposition that evidence once captured or seized must be accounted for from that time onward [10]. Throughout the investigation, analysis and report preparation, the chain-of-custody must be provably kept intact. Feldman [4] recommends that to preserve the chain-of-custody, it is necessary to demonstrate that no information has been added or altered. This can be accomplished by write protecting and virus checking all media, and calculating and signing a message digest.

8. Conclusions

The purpose of our project is to apply forensic principles to a computerized assessment environment to facilitate the identification and prosecution of any party that contravenes assessment regulations. In a con-

trolled assessment environment it is easier to identify any party that contravenes assessment regulations. An institutional policy should permit monitoring of electronic activities, even if this means invasion of the privacy of the learner. The learner should be required to sign an acceptance of the policy and to give consent to be monitored and investigated if a possible contravention is detected.

The utilization of tools for evidence collection and analysis (cross-checking), i.e., key logger, CCTV camera, audit log and report of logins, facilitates the identification of any party that contravenes assessment regulations. The forensic process consists of four phases: preparation of the environment, collection of evidence, analysis of evidence, and reporting findings. The proposed analysis methods are conducted manually, but future research will lead to an automated process. The approach should be applicable to any type of computer-based assessment.

References

[1] I. Armstrong, Computer forensics: Detecting the imprint, *SC Magazine*, August 2002.

[2] M. Bigler, Computer forensics, *Internal Auditor*, vol. 57(1), p. 53, 2000.

[3] C. Boyd and P. Forster, Time and date issues in forensic computing: A case study, *Digital Investigation*, vol. 1(1), pp. 18-23, 2004.

[4] J. Feldman, Collecting and preserving electronic media, Computer Forensics Inc., Seattle, Washington, 2001.

[5] J. Holley, Market survey: Product review, *SC Magazine*, September 2000.

[6] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*, Addison-Wesley Longman, Amsterdam, The Netherlands, 2001.

[7] C. Pfleeger and S. Pfleeger, *Security in Computing*, Prentice Hall, Upper Saddle River, New Jersey, 2003.

[8] S.T. Ltd., Top ten key loggers in review (www.keylogger.org).

[9] W. Soukoreff and S. Mackenzie, KeyCapture (www.dynamicservices.com/~will/academic/textinput/keycapture), 2003.

[10] H. Wolfe, Computer forensics, *Computers and Security*, vol. 22(1), pp. 26-28, 2003.

[11] M. Wyer and S. Eisenbach, LEXIS: An exam invigilation system, *Proceedings of the Fifteenth Conference on Systems Administration*, 2001.

Chapter 10

EXPLORING FORENSIC DATA WITH SELF-ORGANIZING MAPS

B. Fei, J. Eloff, H. Venter and M. Olivier

Abstract This paper discusses the application of a self-organizing map (SOM), an unsupervised learning neural network model, to support decision making by computer forensic investigators and assist them in conducting data analysis in a more efficient manner. A SOM is used to search for patterns in data sets and produce visual displays of the similarities in the data. The paper explores how a SOM can be used as a basis for further analysis. Also, it demonstrates how SOM visualization can provide investigators with greater abilities to interpret and explore data generated by computer forensic tools.

Keywords: Computer forensics, self-organizing map, data visualization

1. Introduction

Dramatic increases in computer-related crimes have led to the development of a slew of forensic tools. These tools ensure that digital evidence is acquired and preserved properly and that the accuracy of results regarding the processing of digital evidence is maintained [9].

Computer forensic investigators are finding it increasingly difficult to use current tools to locate vital evidence in massive volumes of data. In addition, many tools do not present the data in a convenient format for analysis; sometimes, the data presented may actually result in misinforming investigators. In any case, the process of analyzing large volumes of evidence is extremely arduous and time-consuming.

Having an overview of the entire data set obtained directly from a hard drive can be crucial to an investigation. Patterns in the data set could help forensic investigators to locate information, and guide them to the next step in their search.

This paper shows how a self-organizing map (SOM) [6, 7], an unsupervised learning neural network model, can support decision making by computer forensic investigators and assist them in conducting data analysis in a more efficient manner. The technique is used to create graphical representations of large data sets that offer investigators a fresh perspective from which to study the data. In particular, a SOM reveals interesting patterns in data and also serves as a basis for further analysis.

The next section provides background information on computer forensics. The following two sections discuss the SOM technique and its application in computer forensic investigations. The final section, Section 5, presents the conclusions and directions for future work.

2. Background

Computer forensics deals with the preservation, identification, extraction and documentation of digital evidence [9]. Child pornography, threatening e-mails, fraud, and intellectual property theft are all crimes that leave digital tracks [8].

Numerous forensic tools have been developed to collect and/or analyze electronic evidence. Examples include EnCase [5], Forensic Toolkit [1] and ProDiscover [11]. Some tools are designed with a single purpose in mind. Others offer a whole range of functionalities, e.g., advanced searching capabilities, hashing verification and report generation.

A typical computer investigation involves making an exact copy of all the data on a storage medium (e.g., hard drive, compact disk, floppy disk or flash disk). The copy is called an image and the process of making an image is referred to as "imaging." Once the imaging process has been completed, it is essential to have a mechanism or procedure to ensure the integrity [4] of the evidence. Next, it is necessary to analyze the evidence, e.g., performing keyword searches [10] or analyzing signatures and hash values [2].

Computer forensic tools have advanced from using command-line environments to providing sophisticated graphical user interfaces that significantly enhance investigative activities. One useful feature is the presentation of files in a spreadsheet-style format. This ability allows investigators to view all the files on a particular storage medium as well as information regarding each file. The details include file name, file creation date and time, logical size, etc. However, when working with large data sets, the process of scrolling through many rows of data can be extremely tedious. Also, it can be difficult to locate specific information of interest to the investigation.

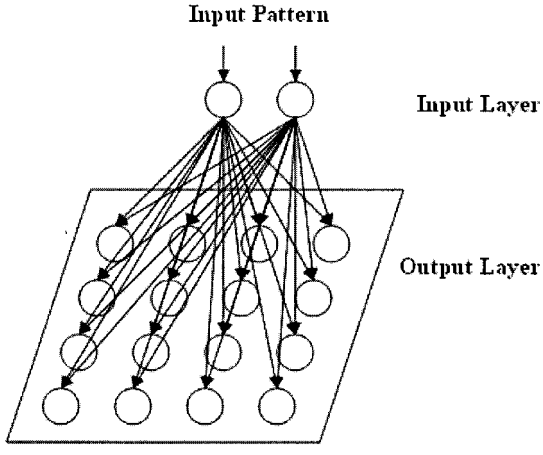


Figure 1. Self-organizing map.

The following section provides a brief overview of a self-organizing map (SOM). The SOM technique is used to enable investigators to visualize all the files on a storage medium and assist them in locating information of interest, both quickly and efficiently.

3. Self-Organizing Map

A self-organizing map (SOM) [6, 7] is a neural network model that has been successfully applied to clustering and visualizing high-dimensional data. It is used to map high-dimensional data onto a low-dimensional space (typically two dimensions). A SOM consists of two layers of neurons or nodes, the input layer and the output layer (Figure 1). The input layer is fully connected with neurons in the output layer and each neuron in the input layer receives an input signal. The output layer generally forms a two-dimensional grid of neurons where each neuron represents a node in the final structure. The connections between neuronal layers are represented by weights whose values represent the strengths of the connections. A SOM is based on unsupervised, competitive learning, which means that the learning process is entirely data driven and that neurons in the output layer compete with one another.

During the learning process, when an input pattern is presented to the input layer, the neurons in the output layer compete with one another. The winning neuron is the one whose weights are the closest to the input pattern in terms of its Euclidian distance [3]. Once the winning neuron has been determined, the weights of the winning neuron and its neighboring neurons are updated, i.e., they are shifted in the direction

of the input pattern. After the learning process, a SOM configures the output neurons into a topological representation of the original data using a self-organization process [6].

The effect of the learning process is to cluster similar patterns while preserving the topology of the input space. However, in order to visualize the different clusters, an additional step is required to determine the cluster boundaries. Once the cluster boundaries have been determined, a SOM can be referred to as a cluster map. The size of a cluster is the number of nodes allocated to the cluster. One way to determine and visualize the cluster boundaries is to calculate the unified distance matrix (U-matrix) [3]. The U-matrix is a representation of a SOM that visualizes the distances between neurons. Large values in the U-matrix indicate the positions of the cluster boundaries.

A SOM is useful for inspecting possible correlations between dimensions in the input data [12]. This can be achieved via component maps. Each component map visualizes the spread of values of a particular component (or dimension). By comparing component maps with one another, possible correlations are revealed.

4. Applying SOM to Forensic Data

A SOM application employs an unsupervised neural network which is trained using forensic data. Two-dimensional maps, i.e., the cluster map and the different component maps, are displayed as hexagonal grids, each grid being referred to as a unit. The discussion of a SOM implementation is outside the scope of this work.

The requirements of computer investigations differ. For example, in the case of child pornography, an investigation involves examining all the graphical images on the suspect's computer system. In most cases, the data presented by forensic tools still requires investigators to manually examine the presented data and draw conclusions.

Figure 2 presents an example of what a computer forensic tool may present to an investigator – a spreadsheet-style display of all the files on the storage medium. This allows investigators to view all the files on the storage medium and to see the details of each file. The process of scrolling through the many rows of data for a large data set can be extremely tedious. However, by applying a SOM, the data set can be mapped to a two-dimensional space for convenient visualization and analysis.

File Name	Ext	File Type	Category	Cr Date
Windows Media Player.lnk	lnk	Shortcut File	Other	2004/08/23...
Windows Marketplace.url	url	Unknown File...	Unknown	2004/08/24...
Windows Catalog.lnk	lnk	Shortcut File	Other	2004/08/23...
Winamp.lnk	lnk	Shortcut File	Other	2004/08/24...
whyv64p2p.ppt	ppt	PowerPoint 9...	Graphic	2004/09/25...
whver.js	js	Unknown File...	Unknown	2004/05/04...
whutils.js	js	Unknown File...	Unknown	2004/05/04...
whtopic.js	js	Unknown File...	Unknown	2004/05/04...
whtcorn2.gif	gif	GIF File	Graphic	2004/03/30...
whtcorn1.gif	gif	GIF File	Graphic	2004/03/30...

Figure 2. Table view of Forensic Toolkit.

4.1 Child Pornography

This section focuses on the application of a SOM to child pornography investigations, in particular, the analysis of temporary Internet files found on a seized hard drive. Most of the temporary Internet files are "image captures" of sites that the user has visited [9]. Obviously, these files may constitute evidence of illegal activity. In an investigation, a law enforcement agent must locate and examine all the images, discover possible patterns and study the suspect's Internet browsing patterns.

We illustrate the application of the SOM technique on an experimental data set containing 2,640 graphical images. The data set generated by Forensic Toolkit [1], a popular forensic tool, contains the four fields listed below. Note that even if file extensions are modified by the user, the tool is able to detect the correct format of each file.

- File name (used only for file identification).
- File extension.
- File creation time.
- File creation date.

The data provided by Forensic Toolkit (strings) cannot be processed directly by the SOM application. Consequently, the strings are converted to numerical values (see Table 1). Dates and times are converted to the formats, "yyyymmdd" and "hhmm," respectively.

The next step is to process the data set with the SOM application. The cluster maps and component maps produced after the SOM training phase are useful visual aids to the investigator.

Figure 3 presents a sample cluster map. The cluster map reveals groups of similar data (clusters), each displayed using a different color.

Table 1. Numerical values for file extensions.

File Extension	Numerical Value
bmp	1
gif	2
jpg	3
png	4

Since this paper is printed in black and white, the clusters are labeled for ease of reference. Figure 3a presents the labeled cluster map corresponding to the one displayed in Figure 3. Similarly, the labeled maps corresponding to Figures 4 and 5 are shown in Figures 4a and 5a, respectively. The letter B indicates the area is blue, C is cyan, G is green, and Y is yellow.

The cluster map in Figure 3 reveals two clusters, one red (R) and the other cyan (C). The brightness of the color reveals the distance of the unit to the center of gravity, i.e., the map unit that most closely represents the average of all the units within a cluster. Brighter colors indicate longer distances, while darker colors indicate shorter distances to the center of gravity.

Component maps, on the other hand, reveal the variations in values of components (or attributes). The combination of all these components determines cluster formation. An example component map is shown in Figure 4. Blue (B) indicates small values, red (R) indicates large values, and the other colors represent intermediate values. The component map in Figure 4 reveals that all the data with small values for the current attribute are grouped in the top right-hand corner of the map. This is the reason why the same units formed a cluster in the cluster map shown in Figure 3. The component maps should therefore be analyzed in conjunction with their corresponding cluster maps.

Figures 5.1 to 5.4 and Figures 5.1a to 5.4a present the unlabeled and labeled maps generated from the image data set after training the SOM. Figures 5.1 and 5.1a present the cluster maps, while the remaining three sets of figures display the component maps for the three components (file creation date, file creation time and file extension).

Figure 5.1 shows that three clusters were formed within the data set. By examining the cluster map and the component maps, it is evident that clusters are formed based on the time when the files were created. This information can be very useful to an investigator.

Figure 5.2 reveals variations in the file creation dates. Blue (B) indicates small values (older files with earlier creation dates) while red (R) indicates large values (new files). Therefore, the most recent files are displayed in the upper half of the map (green (G), yellow (Y), red (R),

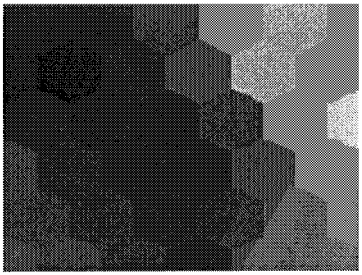


Figure 3. Cluster map.

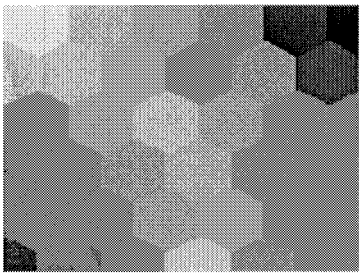


Figure 4. Component map.

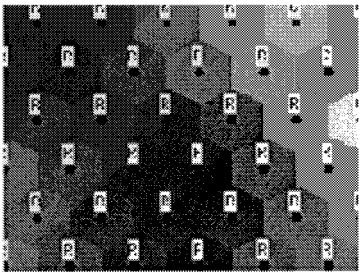


Figure 3a. Labeled cluster map.

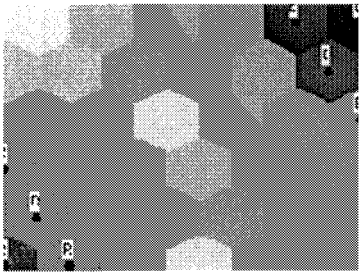


Figure 4a. Labeled component map.

etc.) The bottom half of the map reveals the files that were created earlier (blue (B)). Investigators are thus able to locate the older files by analyzing the bottom portion of the map (see Figure 6).

The older files constitute the area of interest in Figure 6. The specific area being examined by the investigator is marked with a yellow circle. The files were created on 2004/07/31 and each pattern number refers to a particular file or graphical image. This information appears in the bottom right-hand corner of Figure 6. Comparing Figures 6 and 7 confirms that the top portion of the map indeed reflects the files that were created more recently. According to Figure 7, the dates on which the files were created ranged from 2004/10/03 to 2004/11/21.

Correlations are revealed by comparing component maps. For example, comparing Figures 5.2 and 5.3 shows that a correlation exists between file creation dates and file creation times. Most of the recent files were created between 7:00 and 23:59, meaning that the majority of recent Internet activity took place during this time period. Also, simply by examining Figure 5.3, it is possible to discern a downloading behavior pattern. Specifically, most of the images were created between 7:00 and 23:59, corresponding to normal waking hours.

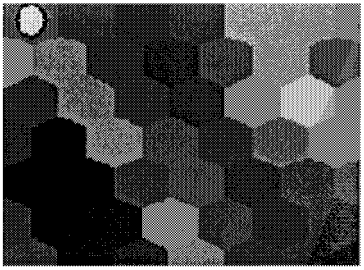


Figure 5.1 Cluster map.

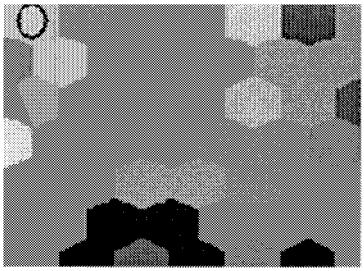


Figure 5.2 Component map (file creation date).

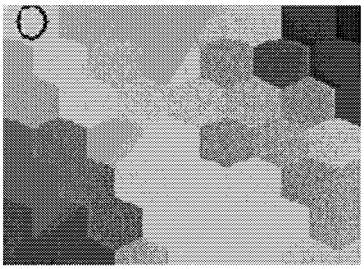


Figure 5.3 Component map (file creation time).

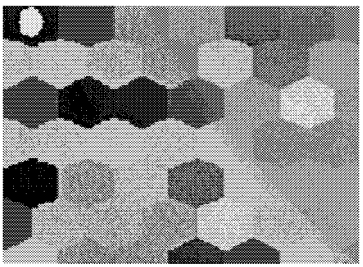


Figure 5.4 Component map (file extension).

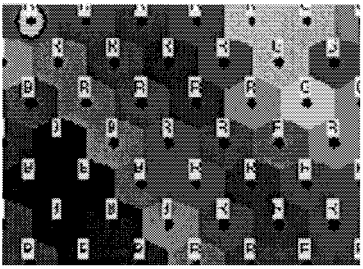


Figure 5.1a Labeled cluster map.

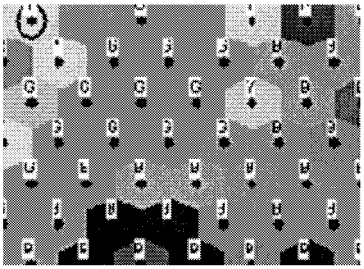


Figure 5.2a Labeled component map (file creation date).

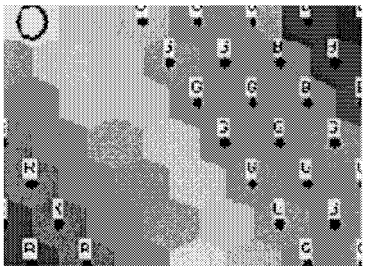


Figure 5.3a Labeled component map (file creation time).

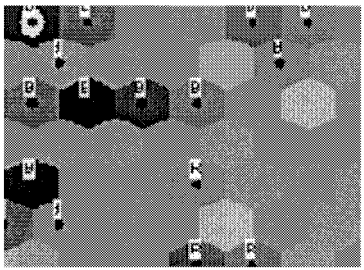
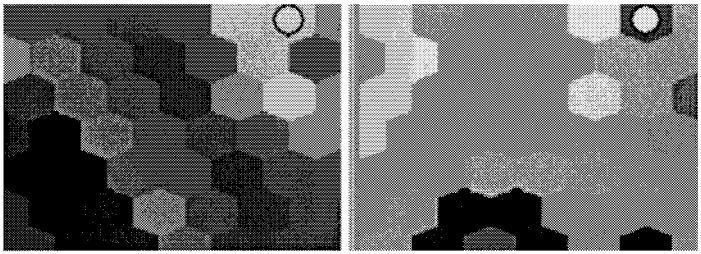


Figure 5.4a Labeled component map (file extension).

	Extension of thi	Time Created	Date Created
Pattern 95	2	1958	20040731
Pattern 117	2	1951	20040731
Pattern 227	2	1955	20040731
Pattern 451	2	1958	20040731
Pattern 649	2	1951	20040731
Pattern 810	2	1958	20040731
Pattern 821	2	1959	20040731

Figure 6. Examining the lower portion of the component map.



	Extension of thi	Time Created	Date Created
Pattern 159	2	157	20041120
Pattern 173	2	152	20041121
Pattern 174	2	152	20041121
Pattern 175	2	140	20041120
Pattern 226	2	226	20041121
Pattern 308	2	202	20041120
Pattern 354	2	232	20041120

Figure 7. Examining the top portion of the component map.

4.2 MP3 Downloading

Another application of a SOM is the investigation of illegal downloading of MP3 (music) files. Downloading MP3 files may be deemed suspicious when large numbers of MP3 files are downloaded in a short period of time. Investigators can discover illegal MP3 downloading patterns using a SOM. In addition, investigators can identify the MP3 files that were downloaded during a certain period of time. By comparing the different maps generated by a SOM application, investigators can determine exactly when large numbers of MP3 files were downloaded. They can also determine the downloading patterns, e.g., every Friday night or once a month at a certain time.

5. Conclusions

A self-organizing map (SOM) can serve as the basis for further analysis of data generated by computer forensic tools. In particular, maps generated by a SOM application create excellent visualizations of large higher-dimensional data sets. These visualizations enable forensic investigators to locate information that is of interest both rapidly and efficiently.

The SOM technique has several applications in digital forensics. These include identifying correlations (associations) in data, discovering and sorting data into groups based on similarity (classification), locating and visually presenting groups of latent facts (clustering), and discovering patterns in data that may lead to useful predictions (forecasting). By providing new perspectives for viewing data, these applications can facilitate the analysis of large data sets encountered in digital forensic investigations. A major drawback, however, is that the data needs to be transformed manually before it can be processed by a SOM application. One avenue for future research is to develop automated data transformation techniques. Another is to design specialized SOM applications for all the major digital forensic processes.

References

[1] AccessData (www.accessdata.com).

[2] E. Casey, *Handbook of Computer Crime Investigation: Forensic Tools and Technology*, Academic Press, San Diego, California, 2002.

[3] A. Engelbrecht, *Computational Intelligence: An Introduction*, Wiley, New York, 2002.

[4] D. Gollman, *Computer Security*, Wiley, New York, 1999.

[5] Guidance Software (www.guidancesoftware.com).

[6] T. Kohonen, The self-organizing map, *Proceedings of the IEEE*, vol. 78(9), pp. 1464-1480, 1990.

[7] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin, Germany, 1995.

[8] W. Kruse and J Heiser, *Computer Forensics: Incident Response Essentials*, Addison-Wesley, Reading, Massachusetts, 2002.

[9] A. Marcella and R. Greenfield (Eds.), *Cyber Forensics: A Field Manual for Collecting, Examining and Preserving Evidence of Computer Crimes*, Auerbach, Boca Raton, Florida, 2002.

[10] D. Schweitzer, *Incident Response: Computer Forensics Toolkit*, Wiley, New York, 2003.

[11] Technology Pathways (www.techpathways.com).

[12] J. Vesanto, SOM-based data visualization methods, *Intelligent Data Analysis*, vol. 3(2), pp. 111-126, 1999.

III

NETWORK FORENSICS

Chapter 11

INTEGRATING DIGITAL FORENSICS IN NETWORK INFRASTRUCTURES

Kulesh Shanmugasundaram, Hervé Brönnimann and Nasir Memon

Abstract This paper explores the idea of integrating digital forensic capabilities into network infrastructures. Building a forensic system for wide area networks has generally been considered infeasible due to the large volume of data that potentially has to be processed and stored. However, it is opportune to revisit this problem in the light of recent advances in data streaming algorithms, the abundance of cheap storage and compute power and, more importantly, increased threats faced by networked infrastructures. This paper discusses the challenges involved in building reliable forensic systems for wide area networks, including the Internet itself. Also, it describes a prototype network forensic system that is currently under development.

Keywords: Network forensics, wide area networks

1. Introduction

The Internet was created using simple rules to serve the communications needs of a well-defined community. During the past ten years, however, it has transformed itself to accommodate a much wider and varied community of users and services with conflicting interests. Although the benefits of a forensic system on a critical infrastructure are obvious, thus far no attempts have been made to incorporate forensic capabilities within the Internet. One reason is the technical difficulty involved in handling large volumes of data on wide area networks. Another is the lack of economic incentives for network operators to deploy and maintain forensic systems. Hence, state-of-the-art network forensic systems simply incorporate augmented packet capture tools for collecting data and *ad hoc* analysis tools for dissecting data. This approach has many inherent problems. Besides being an economic burden on net-

work operators, such systems cannot share data across networks and are, therefore, not very effective. They also lack explicit support for user privacy. Therefore, few incentives exist for network operators to deploy and maintain such systems.

We believe that the lack of forensic capabilities in networks can no longer be ignored; it is the right time to address the problem with forethought and scientific rigor. Advances in streaming algorithms [1] and the availability of cheap storage and computing power can be leveraged to handle large volumes of data. Renewed threats on network infrastructures, increases in cyber crime and litigation in which operators and service providers are held liable for damages are creating the business case for network forensic systems.

This paper presents an approach to implementing network forensics. Also, it discusses the challenges involved in building reliable forensic systems for wide area networks, including the Internet itself.

1.1 System Overview

We are currently building a prototype system to support wide area network forensics. We begin by briefly describing a simple use case of the system. We envision each network domain to have a monitoring policy, privacy policy, and a forensic server associated with it. Further, we assume that network components, e.g., routers and switches, are instrumented to collect data pertaining to various network events. A monitoring policy for the domain describes the events that are monitored and the data collected by the system. The privacy policy for the domain describes access control information on the data. A forensic server is responsible for enforcing the monitoring policy in cooperation with network components in its domain and forensic servers in neighboring domains. It is also responsible for sharing information with appropriate entities in accordance with the privacy policy.

When a user connects to a network, the monitoring and privacy policies are presented to the user, say, via a DNS-like query from the application layer. The policies inform the user of the data collected by the network and how the data are shared. As the user utilizes various networks (enterprise LANs, WLANs, ISP networks, etc.) that form the Internet, each network component collects and records data according to its domain's monitoring policy. The data are either stored within the network components or are transferred to forensic servers for archiving.

When an investigation is launched, e.g., to trace an attack, or even to verify a simple transaction like establishing a connection, an analyst queries the network for relevant information. The forensic servers route

these queries to the appropriate domains, verify the authenticity of the queries, and return the relevant information. For example, suppose the analyst suspects that a malicious connection has been established from a neighboring network, he/she can query the network to assert that it is, indeed, originating from the network and is not being spoofed. Ultimately, if the analyst wants specific details or the connection history of the remote host he/she may have to present an electronic subpoena as defined in the domain's privacy policy. Upon verifying the subpoena, the forensic server reveals the relevant information to the analyst.

1.2 Applications

A network forensic system has several useful applications which are highlighted below.

- **Forensics:** The primary application of a reliable network forensic system is for the *post mortem* analysis of security incidents, especially when host logs are compromised and deemed useless. Compared to the *ad hoc* methods in use today, having a systematic method to collect data across networks would provide reliable evidence as well as valuable corroborating evidence from various vantage points on the network. It would assist in answering questions like: Where did this virus first appear in the network? When and how was confidential information stolen? How did this malware arrive on this host?

- **Network Management and Security:** Forensic data is very valuable for billing, provisioning and making various network management decisions. It can also play an important role in network security. Intrusion detection systems (IDs), for instance, are not effective against slow attacks because they have narrow windows of history. The ability of networks to retain historical event data can enhance the effectiveness of IDSs by providing the contextual information needed to detect slow attacks. For example, IDSs would be able to better detect distributed slow port scans because they can use the forensic system to gather historical data and statistics from their network as well as from neighboring networks.

- **Compliance:** Many regulations (e.g., Sarbanes-Oxley, HIPAA, etc.) and standards (e.g., International Security Standard ISO 17799) pertaining to information systems require that comprehensive audit data be maintained. Obviously, the presence of a reliable network forensic system would significantly support compliance-related activities. The integration of monitoring policies into net-

work infrastructures facilitates the verification of compliance that, in turn, allows users to make their own assessments of the trustworthiness of systems.

In the following subsection, we examine existing solutions to highlight what is missing and what is needed to implement a robust network forensic system.

1.3 Current Solutions and Needs

At present, collecting forensic data is at best an *ad hoc* process. Most networks do not have any means of collecting forensic data aside from firewall logs and intrusion detection logs. The problem is that the scope of the data collected is too narrow, and novel attacks and network abuses may go unnoticed.

There are, however, some innovative solutions that capture and record traffic passing through concentration points in networks [4, 5, 9]. The idea is that the captured data would be available for *post mortem* analysis if needed. Since these tools usually sniff traffic at a single point in a network, they lack the advantages of having multiple views from different vantage points, which would, for example, help spot spoofing attacks. Network Flight Recorder [8] uses distributed sensors to capture and record events. Most solutions, however, take the brute force approach of simply recording raw network traffic. On large networks the amount of traffic would make such solutions infeasible. Furthermore, storing raw data reduces the longevity of data as the storage requirements can be overwhelming, which in turn limits how far back investigations can go. Existing tools do not share the stored information across domains nor do they address privacy issues. Therefore, their deployment has been limited to small LANs, and they do not scale well for large, heterogeneous networks like the Internet. Although distributed intrusion detection systems [6, 7, 13] and traffic measurement systems [2] can be adapted to collect audit data, they lack many of the features required by a network forensic system. Finally, as these solutions augment the existing infrastructure, they add a burden on network operators by incorporating extra hardware and software.

What is needed is a forensic solution that is seamlessly integrated into the very fabric of a network and is transparent to network operators and users, which would make it easy to deploy, maintain and use. Integrating resource-intensive data collection into network infrastructures is challenging as it has to make do with limited memory and processing power. However, a network forensic system is only as effective as the comprehensiveness and longevity of the data it collects. This raises sev-

eral important questions: What events should the forensic system keep track of? How much storage will it take? How much of the existing infrastructure can be reused? What new functionality is expected of networks? How will the added functionality affect routine operations? How should the system handle privacy and security?

2. Research Challenges

An effective network forensic system should monitor and record numerous events on a network to facilitate end-to-end pictures of these events in both macroscopic and microscopic forms. For example, suppose we are interested in the connections between two hosts during some time interval. A macroscopic view of these connections would provide details about the number of connections, the average lifetime of connections and the total amount of data transferred. A microscopic view, on the other hand, would provide details about the individual connections between the hosts, e.g., lifetime of each connection, amount of data transferred, size of individual packets, type of content (audio, text, etc.) associated with each connection and, perhaps, the contents of the connections as well.

This raises an interesting question. How is a network forensic system different from approaches proposed for embedding measurement capabilities in network infrastructures? The difference is that a network forensic system is more general in its purpose and in the information that can be inferred from it (of course, a system that provides fine-grained information can be used for measurements as well). Furthermore, a network forensic system functions without or with only minimal *a priori* knowledge on how the collected data will be used: Will it be used for billing (macroscopic view)? Or will it be used to solve a crime (microscopic view)? Clearly, the data collected by a network forensic system covers a much wider spectrum of events than that collected by a measurement system. Consequently, a network forensic system is more general than existing systems. Much work needs to be done to realize this vision. The following sections discuss some of the challenges that lie ahead.

2.1 What to Collect?

What kind of "events" should a network forensic system track? Ideally, the data collected should help determine the exact path and payload of an arbitrary network packet in the distant past. But this is the ideal case. The challenge then lies in identifying important events and keeping track of these events in an efficient manner. Although the Internet is built on stateless IP, applications and protocols make it a stateful

network. A network forensic system must, therefore, keep track of state changes in this network and its associated entities over time. Information required to keep track of these states is grouped into two categories.

- **Network Dynamics:** The Internet is a very dynamic environment. Different vantage points have different views of the same set of networks and these views may also depend on the protocol levels from which they are viewed. For example, a DNS request for some arbitrary domain may result in two different IP addresses depending on the geographic location of the request and the time of request. Also, two neighboring networks during working days may not be neighbors at nights and weekends due to peering agreements among service providers. The network dynamics are the result of mappings used by the protocols (ARP, DNS, BGP, OSPF, etc.) that glue thousands of networks to form the Internet. A network forensic system must keep track of all this information and use the appropriate information to process forensic queries. For example, it would be incorrect to use current DNS entries to resolve the domain name of an event from the previous month.

- **Traffic Dynamics:** A network forensic system must keep track of information about billions of packets. This information, which is useful for providing macroscopic and microscopic views, includes various global quantities and network traffic trends, packet counts and/or volume broken down by categories (service, source, destination), packet headers, content types, and the payloads themselves.

2.2 How to Store?

Obviously, keeping track of all this information requires massive storage even by present standards. For instance, collecting 64-byte headers of all packets traversing a partial OC3 line in a moderately large network requires 100 GB of storage a day! Therefore, the next logical question is how should the collected data be stored? One approach is to create a "synopsis" of data [12]. A synopsis is a compact representation of the underlying data that is easy to compute. An ideal synopsis would not have any information loss. However, a good synopsis would have a small memory footprint and still capture enough information to be useful for forensic investigations.

Recent advances in streaming algorithms provide mechanisms for creating synposes. Measurement algorithms can be used to develop synopses, but the information they capture is not of sufficient granularity to support forensic queries. Therefore, synopses used in network forensic systems call for a different breed of algorithms. Several streaming

algorithms have been proposed for measuring traffic flow properties in networks. Most of the algorithms provide answers to only a subset of flows, known as "heavy hitters." A network forensic system, however, needs to keep track of all flows, not just heavy hitters. Therefore, these algorithms cannot be used in a network forensic system. A suitable algorithm for collecting traffic dynamics employs space-code bloom filters [3], which help keep track of all flows in a memory-efficient manner.

2.3 How to Retrieve?

Data on wide area networks are widely dispersed. How should the network forensic system locate the data needed for analysis? How could the data be transported efficiently across networks? How should the system present this information to an investigator? The details of the types of data being collected by different nodes in a network, what they collect and their location should be transparent to the investigator. To meet these challenges, novel protocols and distributed query processing mechanisms must be developed to permit the appropriate information to be located within a reasonable amount of time. Moreover, the network must be aware of the privacy requirements of network domains and make sure that these requirements are met. Note that standard relational databases cannot be used because different types of events may be captured in different synopses. For example, TCP connection establishment (connection records) could be stored as a table whereas traffic characteristics could be stored in a tree-based histogram. Even simple arithmetic operations can differ from one synopsis to another. This requires the development of novel database management systems for handling a variety of data types transparently. Since the information presented by the system can be overwhelming for an investigator, appropriate data analysis and visualization tools must be developed.

2.4 Infrastructure Issues

Changing a network infrastructure to accommodate a forensic system presents a set of challenges on its own. How will this system scale on the Internet? What about deployment and coverage? On large networks, e.g., national defense networks and transcontinental private networks, deployment and coverage are relatively easy as there is centralized administrative control. The Internet, however, presents major challenges in terms of deployment and coverage. Still, we believe that even with minimum adaptation to legacy systems, a network forensic system can facilitate the capture of valuable information. For example, if all the upstream paths to a rogue network are covered, it is still possible to in-

fer substantial information about the network using macroscopic views. Support for incremental deployment is another key factor as the adaptation of a network forensic system could go on for several years.

2.5 Privacy and Security

How should a network forensic system support user privacy? How can anonymity on networks be maintained? Network users may have their own privacy requirements (in addition to the prevailing privacy policies) and the network infrastructure should be attentive to these user requirements. Privacy-aware routing protocols must be developed and deployed to assist privacy negotiations in networks. Furthermore, policy monitoring must be integrated into the existing network infrastructure so that user applications can perceive it as being an integral part of the network. Currently, we plan to extend DNS to support monitoring policies. However, privacy negotiations and privacy-aware routing are open questions.

How can the network forensic system be made secure? How should trust be established among nodes? How can the risk of system compromise be minimized? The hierarchical nature of the Internet makes it possible to establish trust using certificates. Proper access control mechanisms and compartmentalization can minimize the risk of system compromise. An interesting research problem is to develop protocols and algorithms that enable network components to collectively decide on the types of events they should monitor to minimize exposure.

2.6 Rethinking Existing Solutions

Once network forensic systems become widely available it would be necessary to rethink many existing solutions so that they can take advantage of the wealth of information available in networks. For example, IDSs can be made more effective by using data collected by a network forensic system. Routing protocols can use historic data to improve their services, and network measurements will be more accurate because data can be gathered from multiple points in a network.

3. The ForNet System

This section describes the architecture of ForNet [12], a network forensic system that articulates the vision outlined in this paper. ForNet has two major functional components: (i) SynApp, which is integrated into networking components (e.g., routers and switches) to provide a secure, scalable and modular environment for collecting data, and (ii) ForensicServer, which functions as a centralized administrative controller for the

domain. A forensic-server is comparable to a DNS server; in fact, we propose to extend DNS to support network forensics.

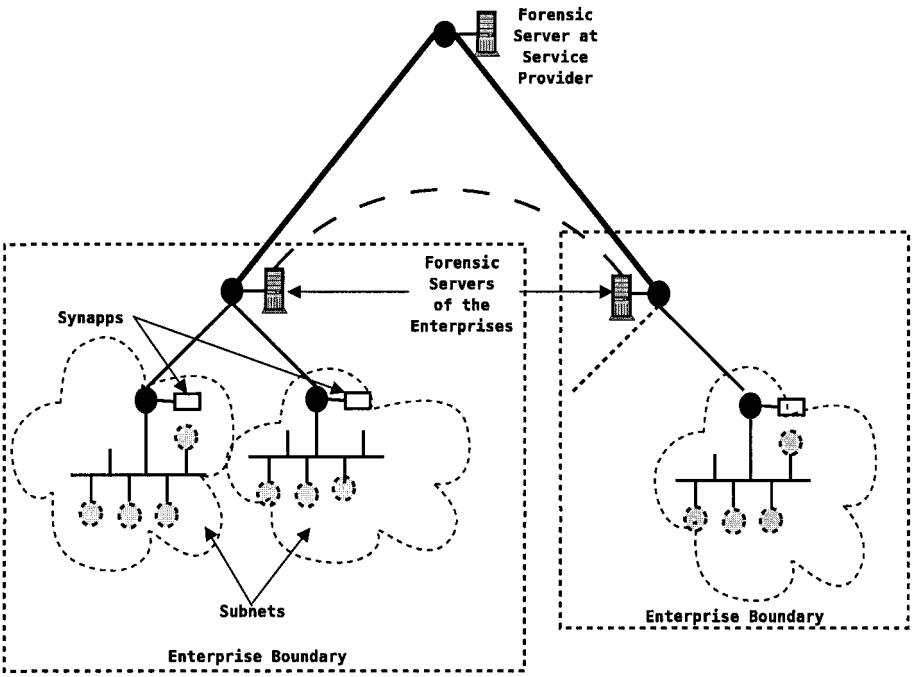


Figure 1. Hierarchical architecture of ForNet with forensic-servers in each network domain and networking components with embedded SynApps.

3.1 System Architecture

The network components instrumented with SynApps and forensic-servers are interconnected in a hierarchy (Figure 1). All SynApps within a domain form a network and are associated with the forensic-server of the domain. The forensic-server receives queries from outside domain boundaries, authenticates the requests and passes them on to appropriate SynApps for processing. Also, it certifies the responses from the agents and send them back to the originators. In addition, the forensic-server advertises the monitoring and privacy policies of the domain and enforces them by verifying that the SynApps and their responses adhere to the policies. Finally, the forensic-server functions as an archiver, receiving collected data from SynApps and storing them for prolonged periods of time (Figure 2).

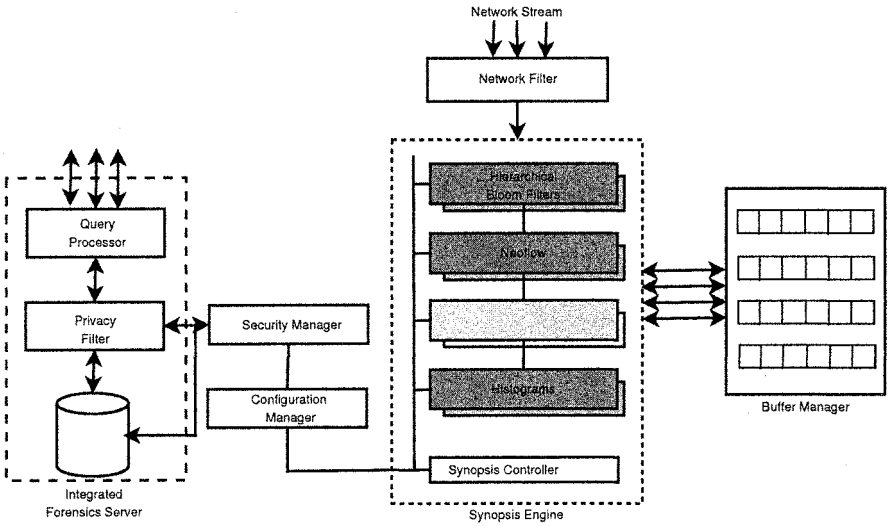


Figure 2. Architecture of a SynApp with an integrated forensics-server.

3.2 Data Architecture

As discussed earlier, data must be collected to facilitate macroscopic and microscopic views. Note, however, that creating microscopic views at the core of Internet is a resource-intensive task due to high traffic volumes. ForNet leverages the hierarchical nature of the Internet to solve this problem. Specifically, SynApps at the leaves of the hierarchy collect fine-grained information necessary for microscopic views whereas SynApps closer to the core collect coarse information necessary for macroscopic views. For instance, a switch at a subnet collects packet content details, an upstream router collects only connection records and an edge router at a service provider collects simple traffic statistics.

This so-called "cascading data collection" has two advantages. First, the data collection workload is spread across the network: SynApps closer to the core only collect macroscopic data and are not overwhelmed by the added functionality. Second, the privacy of users is maintained to a certain extent as SynApps outside a network can only generate macroscopic views and rely on SynApps within the network for detailed microscopic views. This feature naturally helps reduce the amount of storage space required at the network components closer to the core of the network as the traffic volumes are much higher than that at the leaves of the hierarchy.

```
<request>
  <command name="query" scope="module"
      module="ConnectionRecord"
  <event>
    <aspect key="StartTime">X</aspect>
    <aspect key="EndTime">Y</aspect>
  </event>
</request>
```

Figure 3. Query for connections between time X and time Y.

```
<request>
  <command name="query" scope="module"
      module="ConnectionRecord">
  <event>
    <aspect key="StartTime">X</aspect>
    <aspect key="EndTime">Y</aspect>
    <aspect key="DestinationPort">445</aspect>
  </event>
</request>
```

Figure 4. Query for events where destination port is 445.

3.3 Data Structures

Collecting data in a hierarchical manner balances the workload but it does not alleviate storage requirements. A set of synopses is used to reduce the amount of stored data. ForNet uses synopses for capturing compact payload digests in a manner outlined in [10] and for identifying content-types of flows as described in [11]. In addition, ForNet keeps track of connection records, histograms of various traffic statistics, and DNS resolutions. The modular design of SynApp allows the loading and unloading of synopses as necessary. This, in turn, permits cascading data collection in the network. For example, the switches or routers in the subnet level create extended connection records (approx. 30 bytes per record) and require stateful inspection of packets. The routers at the edge create simple connection records with source, destination ports and IP addresses (approx. 12 bytes per record). Upstream routers, on the other hand, uses bloom filters to store simple connection records, which reduce the storage requirement to about 2 bytes per record.

Clearly, cascading data collection reduces the storage requirements at different levels of the network hierarchy while preserving forensic information. Note also that the coarseness of data increases as the network core is approached. This helps strike a balance between security and privacy as it is difficult (if not impossible) to gain meaningful correlations without using information stored at the leaves of a network hierarchy.

3.4 Query Processing and Routing

Cascading data collection results in distributing the collected data evenly across networks. This approach necessitates a reliable query mechanism that can "crawl" a network for data relevant to an event or an investigation. In the ForNet system, a query is defined as a partial description of an event. ForNet uses XML for query expressions and query routing protocols. Figure 3 shows a sample ForNet query for information about all connections during a time interval. Figure 4 shows a more specific query where the destination port is 445.

Of course, we would like to have a query processor where the details of where (which SynApp) and how (in which synopsis) the data are stored are transparent to investigators. ForNet makes query processing transparent using a distributed query processor capable of efficiently locating data within a single SynApp scattered in multiple synopses and in multiple SynApps across networks. ForNet queries can be divided into three major categories: Synopsis Scope Queries, SynApp Scope Queries and Network Scope Queries. The "scope" of a query determines where the data lies in the vast network for SynApps.

- **Synopsis Scope Query:** In a Synopsis Scope Query, an investigator specifically asks ForNet to query a particular synopsis in a SynApp and/or set of SynApps. This is the simplest type of query (see Figures 3 and 4).

- **SynApp Scope Query:** A SynApp Scope Query is produced when an investigator is not sure which synopses must be queried. Instead, he/she sends a general query to a SynApp. The query processor generates a query plan which describes the synopses required to answer the query, the order of chaining of the synopses, and the appropriate merge order of query results from synopses. Then, the query processor executes this query plan and returns the results to the investigator.

- **Network Scope Query:** A Network Scope Query is produced when an investigator wants ForNet to crawl multiple networks for a particular event. Network Scope queries are useful for correlating various events.

During an investigation, queries from a network or set of networks are sent to the appropriate forensic-servers to retrieve evidence. A forensic query is simply a description of one or more events in a set of networks within a certain time interval. A query may describe an event partially and request that the details be filled in by the network. A

```
<request>
  <event>
    <aspect key="payload">0xCAFEBABE</aspect>
  </event>
  <select>
    <aspect key="src" type="ip"/>
    <aspect key="dst" type="ip"/>
    <aspect key="src" type="port"/>
    <aspect key="dst" type="port"/>
    <aspect key="any"/>
  </select>
</request>
```

Figure 5. Network scope query for payload 0xCAFEBABE.

query may be sent to a forensic-server of a network or it may be propagated to forensic-servers in neighboring networks to gather additional information. Queries that cross domain boundaries must go through forensic-servers.

Note that the route a query takes depends on the evidence being sought and is independent of the network topology. At each hop (in this case, a hop is one forensic-server), the query is evaluated and the results are returned. Then, the query moves to the next hop as determined by the results. This propagates until the query terminates within another network. In practice, queries generally begin in one enterprise network and terminate at another enterprise network. In between, the queries travel through all the forensic-servers in the hierarchy until the two endpoints are linked.

Figure 5 presents a sample ForNet query that asks for the source IP, destination IP, port numbers and "any" related information on packets that carried the string "0xCAFEBABE." Note that an "event" is not necessarily related to network changes. Rather, it is an event of interest to an investigator; in this case, it is the occurrence of string "0xCAFEBABE." Readers are referred to [10] for a discussion of similar queries that were used to trace the MyDoom virus in a campus network.

4. Conclusions

This paper deals with the issue of integrating forensic capabilities into network infrastructures. It discusses the research and implementation challenges, and describes an approach for addressing the problem. The prototype system, ForNet, which is currently under development, incorporates novel strategies such as synopses and cascading data collection. The applications of synopses to tracing payloads and detecting network abuses demonstrate its potential for wide area network forensics.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom, Models and issues in data stream systems, *Proceedings of the ACM Symposium on Principles of Database Systems*, 2002.

[2] C. Cranor, T. Johnson, O. Spataschek and V. Shkapenyuk, Gigascope: A stream database for network applications, *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2003.

[3] A. Kumar, L. Li and J. Wang, Space-code bloom filter for efficient traffic flow measurement, *Proceedings of the Internet Measurement Conference*, pp. 167-172, 2003.

[4] A. Mitchell and G. Vigna, Mnemosyne: Designing and implementing network short-term memory, *Proceedings of the International Conference on Engineering of Complex Computer Systems*, 2002.

[5] Network General, Infinistream (www.networkgeneral.com).

[6] V. Paxson, Bro: A system for detecting network intruders in real-time, *Proceedings of the Seventh Annual USENIX Security Symposium*, 1998.

[7] P. Porras and P. Neumann, Emerald: Event monitoring enabling responses to anomalous live disturbances, *Proceedings of the National Information Systems Security Conference*, 1997.

[8] M. Ranum, K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth and E. Wal, Implementing a generalized tool for network monitoring, *Proceedings of the Eleventh Systems Administration Conference*, 1997.

[9] Sandstorm Enterprises, Netintercept (www.sandstorm.net).

[10] K. Shanmugasundaram, H. Bronnimann and N. Memon, Payload attribution via hierarchical bloom filters, ISIS Technical Report, Polytechnic University, Brooklyn, New York, 2004.

[11] K. Shanmugasundaram, M. Kharazi and N. Memon, Nabs: A system for detecting resource abuses via characterization of flow content type, *Proceedings of the Annual Computer Security Applications Conference*, 2004.

[12] 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.

[13] V. Yegneswaran, P. Barford and S. Jha, Global intrusion detection in the DOMINO overlay system, *Proceedings of the Network and Distributed System Security Symposium*, 2004.

Chapter 12

USING PEER-TO-PEER TECHNOLOGY FOR NETWORK FORENSICS

Scott Redding

Abstract Networked computer systems are under a barrage by combatants attempting to obtain unauthorized access to their resources. Methods must be developed to identify attacks on the systems and provide a forensically accurate description of the chain of events related to the unauthorized activity. This paper proposes a peer-to-peer (P2P) framework for network monitoring and forensics. Host-based security tools can be used to identify malicious events. The events can be communicated to other peers over a P2P network, where analysis, forensic preservation, and reporting of related information can be performed using spare CPU cycles.

Keywords: Network forensics, peer-to-peer networks

1. Introduction

Networked computer systems are under a barrage by combatants attempting to obtain unauthorized access to their computing resources. Methods must be developed to identify attacks on the systems and provide forensic evidence of the chain of events related to the unauthorized activity. While the ideal solution may be to save a record of all bits that traverse a computer network, that is practically infeasible due to the massive amount of traffic on high speed networks. Therefore, a more reasonable solution is to use end systems on the network to identify the interesting data and to inform other systems of their findings in order to: analyze events, identify attacks, share findings, protect the network, and to preserve evidence. Important data relating to unauthorized access of the systems must be preserved in a forensically acceptable manner while striving to minimize the data storage resources.

2. Peer-to-Peer Framework

The P2P network framework approach to network monitoring and forensics is a solution that utilizes end systems in a peer-to-peer manner to collect and analyze host-based security events. A security event is any incident identified by a host-based protection mechanism. The P2P network forensics architecture is reliant on cooperation from as many peers as possible within a community of interest. A neighborhood or peer group is a set of peers sharing a common attribute. This attribute could be operating system, network interface type, hardware similarity, software application, platform use, server type, network subnet, location, organizational work group, or even an end user characteristic. Each of the systems can be member of many neighborhoods in the P2P network and can effectively contribute data pertaining to the status of their neighborhoods. A community is comprised of the set of all neighborhoods. The P2P framework is based on the Java programming language, so inclusion of a variety of computing devices is easily accomplished.

All peers should be ready to contribute data to the P2P network whenever they are running. Data acquisition is performed through interaction with the existing host-based protection applications running on the peer. Data is shared with other neighborhood systems via the P2P network. On-line peers which are currently inactive, or active less than some threshold, perform analysis of the neighborhood network data that is received. The P2P network forensic system is designed to be able to deal with the transient nature of peers.

Utilization of peers in the collection and analysis of network data is a resourceful and economical use of existing systems. Since a community's infrastructure is comprised of these systems which are already in place, it is a logical step to use them to perform network data analysis. These systems already have access to the network traffic that is of the most interest which is the traffic to the systems themselves. As Denning proposed, the systems also are best able to determine what is legitimate traffic and what is illegitimate or anomalous [4]. Use of the systems themselves as network sensors is worthwhile as long as that task doesn't interfere with the official mission of the system. In addition to the ability of peer systems to capture relevant traffic, the amount of idle CPU cycles available on the peer systems during normal processing and especially during times when no active processing is going on which can be used to analyze network information can be significant. As the success that P2P applications such as SETI@HOME [1, 11], the Folding Project [5], and the World Community Grid [16] has shown, using idle

cycles to perform background processing can be an efficient utilization of computing resources.

3. Network Forensics

This work engages the network forensics definition provided by the Digital Forensic Research Workshop. Network forensics is "the use of scientifically proven techniques to collect, fuse, identify, examine, correlate, analyze, and document digital evidence from multiple, actively processing and transmitting digital sources for the purpose of uncovering facts related to the planned intent, or measured success of unauthorized activities meant to disrupt, corrupt, and or compromise system components as well as providing information to assist in response to or recover from these activities" [8].

In order to perform network forensics using P2P technology, it must be shown that the P2P techniques used are scientifically sound and that they can uncover the facts related to unauthorized access of computing resources. The network forensics process can be broken down into five distinct phases. The first phase is event acquisition. An event is any occurrence for which a host-based protection mechanism issues an alert or notice. The P2P application on a peer is responsible for collecting these alerts from the set of host-based protection systems. The event is normalized into a standard XML document format in order to ease the parsing process used by other peers. The next phase is event transmission. This is really where P2P comes into play. The P2P network is designed so that the normalized XML version of the event is transmitted to all other neighbor peers. This is done through the P2P network without any configured or pre-existing information about the identity and location of neighbor peers. The third phase is data storage. Information regarding specific events must be archived in a manner allowing investigations to proceed without the concern that the data describing the event has been corrupted. By creating databases on each of the peers containing event information for all of their neighbors, corroboration of the data can be accomplished. The fourth phase is data analysis. As each peer receives transmission of an event from one of its neighbor peers, it tries to correlate that event with other events that have been archived in its database. After analysis, the final phase, reporting, is invoked. A peer is designed to report its findings back to all neighboring peers in order to create a complete neighborhood view for all peers.

Network security personnel can configure monitor peers that are members of all neighborhoods in order to establish a profile of the complete network status. A peer can also develop a peer status showing each of

the neighborhoods in which it belongs. The techniques used in each of these phases will be described in order to show that they are scientifically sound.

4. Data Acquisition

Data acquisition is the process where host-based protection security mechanisms collect and identify events that can contribute to the network forensic process. This acquisition can be separated into two levels. The first level involves the protection mechanisms that are normally running on the peer. The events from the first level are considered important enough to be transmitted to all other peers belonging to the same neighborhood. The second level is enacted after an event or a series of events has been deemed important enough to warrant an increased amount of effort in data acquisition and an increased volume of resultant data. The first level mechanisms include host-based firewalls, anti-virus systems, and intrusion detection systems. Intrusion detections systems can be further broken down into programs which perform file system monitoring, log file analysis, and network connection monitoring. The second level mechanisms are packet capturing programs.

As an example of a first level data acquisition process, a firewall system will be described. Consider the iptables [9] firewalling product for linux. The iptables program can be configured to log using the syslog daemon any network packet that meets some criteria. These criteria are based on the security policy for the host. Packets can be logged independently of the action of the firewall to accept, reject or drop them. Like many of the host-based protection mechanisms, iptables, has its own log format. The iptables log includes information that could be very useful in the analysis process: IP header fields, TCP header fields, UDP header fields, IP options, TCP options, and TCP sequence number. Each of the log fields is formatted with the field name, an equals character, and the field value except in the case where the field is a boolean flag, and in that case, if the log entry includes the field name alone, it indicates that the flag is set. A sample log entry is:

```
Nov 22 09:55:17 myhost kernel:iptables REJECT IN= OUT=eth0
SRC=192.168.118.221 DST=192.168.1.19 LEN=60 TOS=0x00 PREC=0x00
TTL=64 ID=12320 DF PROTO=TCP SPT=36718 DPT=767 SEQ=890512859
ACK=0 WINDOW=5840 RES=0x00 SYN URGP=0
```

As long as the format of the host-based protection mechanism log entry is known, the entry can be reformatted into an XML document to ease the interpretation by other peers. Converting the information into a

XML document allows each of the peers in the neighborhood to be able to use the XML parsing mechanism built into its P2P application to process the event without having to have a priori knowledge about the specific log format of the host peer's protection program.

The second level of data acquisition occurs after a host-based protection event is determined to be important. This second level is an attempt to gather more information about the event. The analytical process 12.7is used to determine when an event meets the criteria of being important. This second level acquisition process is an attempt at collecting more complete network information. Packet capturing systems such as snort [10, 13], tcpdump [14, 15], or snoop [12] are then utilized to collect and archive relevant data. As opposed to the host-based protection mechanisms, the second level acquisition processes are designed to gather all relevant information. While this involves substantially more data collection, it is still less than the amount of data that would be collected if a capture all method were employed.

5. Communication Architecture

P2P refers to the concept that in a network of systems, any system can communicate or share resources with another without necessarily needing central coordination. Each of the systems (peers) in the network are essentially treated as equals. The common non-P2P method which is prevalent in today's network environment is client-server where the server is a centralized system providing information and coordination to a number of subordinate client systems. P2P networks on the other hand are designed such that all systems are producers as well as consumers. A pure P2P network does not rely on a centralized management system in order to control communication. It is designed to allow dynamic discovery of peer systems and permit direct communication between any of the peers without the need for intervening control. In opposition to client/server networks, P2P networks increase utility with the addition of more nodes. In order to create the distributed network forensics environment, a peer-to-peer network is established. This P2P network allows peer/node/end devices to communicate with each other to share interesting network information.

A P2P system for performing network forensics has the following requirements:

- Must permit peers to be both producers and consumers of data.
- Must allow the dynamic addition and subtraction of peers to the system.
- Must communicate with a variety of platforms.
- Must minimize network traffic overhead.
- Must facilitate the exchange of data in a common format.

- Must ensure confidentiality.
- Must ensure integrity.
- Must ensure availability.

The P2P framework that was found to best satisfy the requirements is JXTA (pronounced juxta) [6].

5.1 JXTA

5.1.1 Overview. The P2P framework is based on the JXTA open source platform from Sun Microsystems. JXTA technology is a Java based network programming and computing platform that is designed to solve a number of problems in modern distributed computing. Project JXTA's objectives are to enable interoperability between platform independent systems. The goal is JXTA ubiquity, implementable on every device with a digital heartbeat, including sensors, consumer electronics, PDAs, appliances, network routers, desktop computers, data-center servers, and storage systems.

5.1.2 Architecture. The JXTA software architecture is referred to by the Protocol Specification (v2.0) as a "three layered cake." The three layers are:

- Platform. This layer encapsulates minimal and essential primitives that are common to P2P networking, including peers, peer groups, discovery, communication, monitoring, and associated security primitives. This layer is ideally shared by all P2P devices so that interoperability becomes possible.

- Services. This layer includes network services that may not be absolutely necessary for a P2P network to operate but are common or desirable for P2P environments. Examples of network services include search and indexing, directory, storage systems, file sharing, distributed file systems, resource aggregation and renting, protocol translation, authentication and PKI services.

- Applications. This layer includes P2P instant messaging, entertainment content management and delivery, P2P e-mail systems, distributed auction systems, and many others. Obviously, the boundary between services and applications is not rigid. An application to one customer can be viewed as a service to another customer.

This system fits into the JXTA applications layer. On top of the underlying JXTA platform a P2P network is designed. Peer groups are a

central concept in the JXTA platform which provides a segmentation of the P2P network space into distinct sets. The P2P framework relies on these JXTA peer groups for implementation of the peer neighborhoods. In this P2P architecture, each peer by default is a member of the Net Peer Group. This essentially is a network where each peer knows about and can set up communication with all other peers. Peer groups are created in order to partition the network into logical neighborhoods so communication can be restricted to only those other peers who have an interest in similar data.

5.1.3 P2P Communication. The P2P framework communication consists of an end system establishing itself in the P2P network and then performing transfer of messages describing interesting network traffic to its peer neighbors. These messages could be periodic messages of average statistics, and then individual messages consisting of interesting data. The messages are of a standard XML based format that any P2P host can parse. Peers also transfer messages related to the ongoing analysis of its data. Through the communication of interesting events and individual analysis of a peer's environment/history a more thorough understanding of an event can be obtained.

5.1.4 Security. As with any computing system or environment, security of the system must be addressed. Using a completely distributed manner for performing the network forensics eliminates a big problem with a centralized server which is that it is a single point of failure. The P2P network forensics analysis technique also provides a great deal of redundancy to the process since any interesting data and any significant analytical results on a peer are likely to be duplicated on other peers in its neighborhood.

The P2P network forensics model presented here is for an administratively closed environment. That is, all systems in the community are contained within a single administrative domain. Given this architecture, a Public Key Infrastructure (PKI) approach to security, specifically trust, can be implemented. Security in an information system can be considered as three legs, confidentiality, integrity, and availability and each of those legs must be addressed.

Confidentiality is the "concealment of information or resources" [2]. Since this is an administratively closed environment model, sharing of the information within the P2P network is allowed. A larger issue is preventing outsiders, that is, those not in the community, from accessing the information. This is an community wide security issue but sill important to the design of the P2P network. If the network boundary is

not sufficient to ensure confidentiality, P2P message traffic within peer neighborhoods can be encrypted using the PKI.

Integrity refers to the "trustworthiness of data or resources, and is usually phrased in terms of preventing improper or unauthorized change" [2]. Integrity is the very important in the P2P architecture. Each peer needs to be confident that the information that it receives from other peers is accurate. This can be assured through the implementation of a PKI. The PKI is used with public key cryptography and digital signatures to verify the integrity of messages between peers. Within the administratively closed environment, a Certificate Authority (CA) is established. The CA's verification of a peer's public key is contained in a digital certificate. Each peer presents its digital certificate to other peers as a means to securely distribute its public key. To prove that a P2P message is authentic, a peer digitally signs the message with its private key and transfers the signed message. The receiving peer can authenticate the message using the public key included in the digital certificate. Integrity is then assured.

Availability refers to the "ability to use the information or resource desired" [2]. The concern here is twofold. First, it is a matter of community wide management. Peers need to be able to get interesting information from the other peers in its neighborhood. This means that other peers must be contributing in the P2P forensic network. Free-loading on P2P networks is a common concern [7], which in this environment can be dealt with administratively. Secondly, the P2P network must have controls in place so that peers are not overwhelmed with P2P messages which could create a denial of service. This must be handled in the P2P application. The application must recognize when it is sending out too many messages, and throttle back. The fact that too many messages are being generated must be taken into account in the self-analysis of the host and treated appropriately. The number of messages sent through the P2P network must be one of the items that is statistically monitored and when a significant deviation from this value is encountered an anomaly event should be triggered.

6. Archival

Two aspects of archival portion must be addressed: archival for audit purposes and archival for analytical purposes. Audit records are required to perform after the fact recreations of the unauthorized activity and are more rigorous in their requirements. These records need to be authentic (trusted). Analytical records on the other hand, are copies of the event that a peer receives from a neighbor and are used in the analytical

process of determining if malicious activity is underway. These records diminish in importance as time progresses and can be summarized or deleted with the presumption that the loss of information will not have significant impact.

Since the event information that is produced from a peer is based on the host-based protection mechanisms from that peer, that application is responsible for archiving the information for audit purposes. The P2P events can be recreated later from the archived information if needed. The P2P process needs to be responsible for archival of the received events from other peers in order to do effective analysis. A means to archive/database the event information needs to be developed. Some of the goals of this portion of the project are minimal data storage requirements, ease of searching the archive for relevant information, and ability to deploy the archive on any platform that the P2P application runs on. The archival system should automatically age the events and take appropriate steps to remove outdated information.

7. Analysis

A critical element of the network forensic process is to examine, correlate, and analyze the digital evidence that is collected and shared using the P2P process. Each peer system in the network is expected to collect the information that its neighbor peers identify as security events and perform correlation of that data with data that the peer has already archived in order to determine if the event is worthy of more detailed scrutiny. Analysis is the process by which an event is determined to be worthy of being elevated to the important category. The first step of analysis actually occurs in the data acquisition phase of the process. This is the normalization of the data. Normalization means to make the data regular and consistent.

The analysis process is triggered each time that an event arrives over the P2P network. Reception of the event causes the analytical process to compare the event with other events that have been archived on the peer. There are many different analytical approaches that can be employed to determine if an event should be elevated to the important level. Two of these approaches are statistical analysis and rule based analysis.

Statistical analysis is a process where the occurrence of an event is compared to what has been statistically determined to be normal use. Denning [4] hypothesized that exploitation of system vulnerabilities involves abnormal use of the system; thus, security violations can be detected by analyzing abnormal patterns of system usage. In particular,

Denning identifies five models by which abnormal behavior may be identified:

Operational Model: Abnormality here is defined as a comparison of an observation against a fixed value. The fixed value is not based on a statistical evaluation of some n previous events, but upon a predetermined threshold which can be based on security policy. An example is an event counter of login failures due to bad password within some time constraint.

Mean and Standard Deviation Model: Abnormality is based on comparison of an event to previous events. A mean and standard deviation of the previous events are calculated and an event is abnormal if it doesn't fall within a confidence interval based on a number of standard deviations from the mean. Chebyshev's inequality tells us that the probability of a value falling outside of that interval is at most the inverse of the number of standard deviations squared.

Multivariate Model: Similar to the mean and standard deviation model except that it is based on the correlation of multiple variables. A study by Ye, *et al.* [17] discusses this model and compares two techniques, Hotelling's T^2 test and the chi-squared distance test, X^2. They find that the more scalable X^2 multivariate analysis technique detecting mean shifts only is sufficient for intrusion detection.

Markov Process Model: Using each individual event as a state, and a state transition matrix to characterize transition frequencies between states, an event can be categorized as abnormal if the probability of transition between the previous state and the event state is low as determined by the state transition matrix.

Time Series Model: Similar to the mean and standard deviation model, but includes the timestamp of the event in its determination of abnormality. In this case an event is considered abnormal if the probability of the event occurring at that time is too low. This method has the advantage of measuring trends over time and detecting behavioral changes.

The rule based analysis approach, which is similar to the Markov Process model, is presented by Chuvakin [3]. According to Chuvakin, rule-based correlation uses some preexisting knowledge of an attack (a rule), which is essentially a scenario that an attack must follow to be detected. Such a scenario might be encoded in the form of "if *this*, then *that*, therefore *some action* is needed."

Rule based analysis deals with: states, conditions, timeouts, and actions. A state is a situation that an event from a peer describes, it is a position that a correlation rule is in. A state is made up of various conditions that describe the state like source or destination IP addresses and port numbers. The timeout describes how long the rule sequence should be in a particular state and a transition is the movement from one state to the next. Rules may describe many different state transitions from a particular state. An action is the steps taken when a rule has been matched.

Unlike many of the statistical analysis methods, rule based analysis requires a understanding of the specific operation of the system, or known attack sequences that occur. The first, specific operation, tracks state changes through normal operation of the system and deviation from the expected rules indicate abnormal activity which could be an indicator of malicious intent. The second, known attacks, tracks state changes though known malicious activities and a complete rule match will be an indicator that the malicious activity has occurred.

An analysis of an event that results in a determination that the event is significant triggers the analyzing peer to send a message to its neighbors that an important event has occurred. As stated previously, when an event is deemed to be important, neighbors are alerted in order for them to collect more detailed information regarding similar events. The message that is sent contains critical parameters based on the event type which will allow the neighbors to determine what to further scrutinize. The message will also be an important factor in alerting the system user and administrator to the fact that a possible unauthorized access of information resources was identified.

8. Reporting

Reporting is the documentation part of network forensics. The analytical results that each peer produces need to be provided to the user/administrator of the peer system. Since the analytical process is designed to run in the background using spare CPU cycles, the best way to display results would be in the form of a screensaver. While this provides information to the user of the peer, overall community results are also necessary. The designed method for this type of operation is to dedicate an analytical/reporting peer or set of peers that operate exclusively for the purpose of performing monitoring. These peers operate in the same manner as normal peers, but will be configured to join any neighborhoods that they are interested in monitoring.

9. Conclusions

A P2P framework used to share security events among information systems in order to perform network forensics has been proposed. The security events that are shared are normalized versions of alerts gathered from existing host-based protection mechanisms. A prototype P2P application using JXTA technology has been developed and shows promise in effectively transferring the information throughout a community comprised of a number of peer neighborhoods.

Future work will focus on employing scientific techniques to collect, fuse, identify, examine, correlate, analyze, and document digital evidence from multiple, actively processing and transmitting digital sources. The ultimate goal is to uncover facts related to the planned intent, or measured success of unauthorized activities meant to disrupt, corrupt, and/or compromise system components as well as providing information to assist in the response and/or recovery from these activities.

References

[1] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky and D. Werthimer, SETI@home, *Communications of the ACM*, vol. 45(11), pp. 56-62, 2002.

[2] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley, Reading, Massachusetts, 2003.

[3] A. Chuvakin, Security event analysis through correlation, *Information Systems Security*, vol. 2(13), pp. 13-18, 2004.

[4] D. Denning, An intrusion-detection model, *IEEE Transactions on Software Engineering*, vol. 13(2), pp. 222-231, 1987.

[5] FOLDING@home (folding.stanford.edu).

[6] JXTA (www.jxta.org).

[7] A. Oram (Ed.), *Peer-To-Peer Harnessing the Power of Disruptive Technologies*, O'Reilly, Sebastopol, California, 2001.

[8] G. Palmer, A road map for digital forensic research, *Proceedings of the Digital Forensic Research Workshop*, 2001.

[9] netfilter/iptables (www.netfilter.org).

[10] M. Roesch, SNORT - Lightweight intrusion detection for networks, *Proceedings of the Thirteenth Systems Administration Conference*, 1999.

[11] SETI@home (setiathome.ssl.berkeley.edu).

[12] snoop (docs.sun.com).

[13] snort (www.snort.org).

[14] R. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley, Reading, Massachusetts, 1994.

[15] tcpdump (www.tcpdump.org).

[16] World Community Grid (www.worldcommunitygrid.org).

[17] N. Ye, S. Emran, Q. Chen and S. Vilbert, Multivariate statistical analysis of audit trails for host-based intrusion detection, *IEEE Transactions on Computers*, vol. 51(7), pp. 810-820, 2002.

Chapter 13

FORENSIC PROFILING SYSTEM

P. Kahai, M. Srinivasan, K. Namuduri and R. Pendse

Abstract Hacking and network intrusion incidents are on the increase. However, a major drawback to identifying and apprehending malicious individuals is the lack of efficient attribution mechanisms. This paper proposes a forensic profiling system that accommodates real-time evidence collection as a network feature to address the difficulties involved in collecting evidence against attackers.

Keywords: Forensic profile, intrusion detection, alert, probe, audit trail

1. Introduction

Most organizations secure their networks using encryption technologies, network monitoring tools, firewalls and intrusion detection and response mechanisms. Despite all these security measures, compromises occur daily. Evidence collection, IP traceback and identification of attackers are as important as effective intrusion detection when a network attack takes place. However, while intrusion detection systems (IDSs) are fairly mature, very few tools exist for IP traceback and attacker identification. Prosecution is hampered by the non-availability of evidence in cases involving expert hackers and jurisdictional constraints on law enforcement. This paper proposes a forensic profiling system that accommodates real-time evidence collection as a network feature to address the difficulties involved in collecting evidence against attackers.

2. Related Work

Collaboration between intrusion detection and response systems has been the focus of recent research. MIRADOR [3] implements cooperation between multiple intrusion detection systems through a cooperation module. The cooperation module, CRIM [4], provides the interface for

alert clustering, alert merging and alert correlation. The Common Intrusion Specification Language (CISL) [7] presents a language for communication between intrusion detection systems in a network.

Alert aggregation and alert correlation have been investigated by several researchers [3–5, 10, 13]. The clustering of similar intrusion alerts is discussed in [3, 13], but the authors do not emphasize the underlying relationships between alerts. Also, most alert correlation methods are restricted to known attack scenarios. A formal framework for alert correlation and detection of multi-stage attacks is described in [10]. Alert correlation is performed if the consequences of a previous alert serve as prerequisites for the current alert. But the alerts do not confirm the possible consequences. For example, the detection of a buffer overflow attack does not imply that the attacker was successful in acquiring root privileges. In order to determine if the attack was indeed successful, participation from other network components is essential. This paper proposes a mechanism for real-time forensic evidence collection, where each network element that detects suspicious activity provides evidence in the form of log entries indicative of the activity.

3. Forensic Profiling System

The Forensic Profiling System (FPS) engages a client-server architecture. Each node in the network, referred to as a *forensic client*, is capable of detecting an anomaly, upon which it warns a central *forensic server* about the anomaly in the form of an alert. All the forensic clients participate in distributed intrusion detection and, therefore, maintain logs. A forensic client can be a router, signature analyzer, IDS, firewall or network host. The FPS logical architecture, presented in Figure 1, shows the interactions between the forensic server and forensic clients using alerts and probes.

Upon detecting an incident, a forensic client sends an alert to the forensic server along with evidence (logs) indicative of the incident. The forensic server correlates the alerts and the responses to any probes it issues and builds a forensic profile. The generation of probes is dependent on the forensic profile database maintained by the forensic server. The database contains information about known/investigated attacks in the form of descriptors called *forensic profiles*.

3.1 Forensic Profiles

A forensic profile is a structure that provides information about an attack in a succinct form. In its nascent state, a forensic profile, is based on knowledge about an attack; it is a collection of alerts that provides

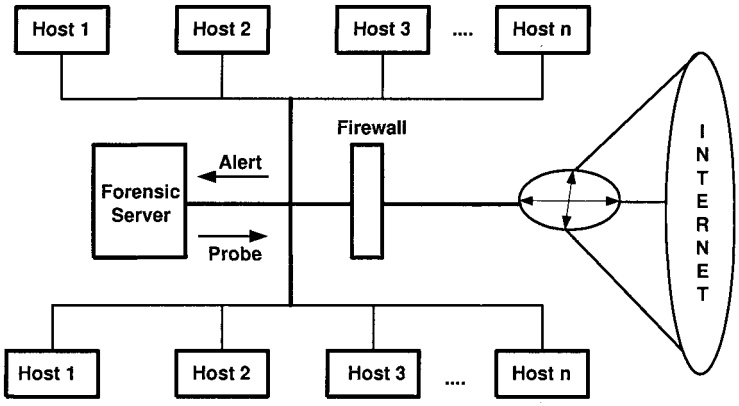


Figure 1. Forensic Profiling System (FPS) architecture.

an indication of the attack. An attack is composed of a series of inter-related events. A subset of these events might be common to several attacks. Thus, a stand-alone event does not give complete information about an attack. In order to ascertain that a particular attack has occurred, a certain minimum number of events must be detected. A profile is a structure/descriptor that defines an attack in terms of its related events; it provides relevant information about the attack in terms of its associated alerts.

The *passive state* of a network is defined as the state wherein the network is involved in normal activities that do not impact its security. In the passive state, the forensic server maintains a database of *passive profiles* of all known attacks. The passive profile is partial because it provides static and general information about an attack. The detection of an event generates an alert. A passive profile may become active when an alert is generated. The passive profiles, which contain a match for the alert generated, are considered to be active. Figure 2 shows the relationship between the Alert X received from a forensic client with the forensic profile database, which is required to shortlist all active profiles. Alert X is a subset of the alerts associated with Forensic Profiles 1 and 3. Therefore, the Profile Descriptors 1 and 3 are activated.

As an alert is a parameter of a profile, the forensic server searches for a match for the alert in the passive profiles and generates a stack of *active profiles.* To select a specific active profile, the forensic server queries the other network entities for a similar kind of event by transmitting a probe to all the forensic clients. If a forensic client responds with information pertinent to one or more active profiles, the forensic server reduces the

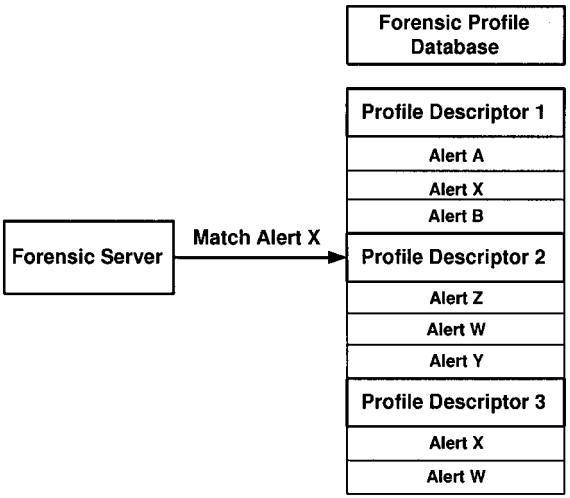


Figure 2. Active profile descriptors created by an alert.

active stack accordingly. This process recurses until the entire attack is detected. The forensic profile is thus constructed from the detection of the first alert to the detection of the attack.

3.2 Client-Server

The forensic server coordinates the activities of forensic clients. A forensic client continuously searches for anomalous activity and listens to probes from the server through agent_alert and agent_probe, respectively. The detection of a security incident involves *active monitoring* and *active parsing*. Active monitoring involves observing performance parameters such as CPU utilization and event-log intensity of the client and checking for discrepancies. Active parsing involves continuously scanning entries in the log files and history files and checking for suspicious entries (keywords), such as authentication failure, access denied and connection failure. An alert has *When-Subject-Object-Action* fields. *Subject* is the forensic client that triggers the alert, *Action* specifies the event, and *Object* is the network element on which the *Action* occurs. Events capable of triggering alerts are listed in Table 1.

The forensic server generates two kinds of probes, Check_Probe and GetLog_Probe. Check_Probe checks for suspicious activity in relation to an earlier alert received by the server. If the forensic client responds with a NULL packet to the Check_Probe then the server does not send a GetLog_Probe. Otherwise, the forensic server sends GetLog_Probe to

Table 1. Suspicious events capable of triggering alerts by forensic clients

Event	Alert
Change in CPU Utilization	*CPU_Util{SubjectIP, Up/Lo Flag, Percent}*
Frequency of log entries	*Log_Intensity{SubjectIP, FreqLogEntries}*
Increased Memory Utilization	*Mem_Util{SubjectIP, Percent}*
N/w utilization	*BandWidth{ SubjectIP, CurrentUtil}*
Login Denied	*DeniedLogin{SubjectIP, Username, Remote/local, AttemptNo}*
Invalid IP (reserved and not used)	*InavalidIP{SubjectIP, Invalid IP}*
System File deletion	*DeleteFile{SubjectIP, FileName}*
Change of privileges for System Log	*Chmod{SubjectIP, Syslog}*
Change of privileges for History File	*Chmod{SubjectIP, Hist}*
Connection Failed	*FailCon{SrcIP:SrcPort,DestIP:DestPort, AttemptNo}*
Upload File	*FileUpload{SubjectIP (Server), FileName}*
Unsetting History File	*UnsetHist{SubjectIP}*
Change attributes of history file	*Chattr{SubjectIP, Hist}*
Kill logging daemon	*Kill{SubjectIP, LogD}*
Kill kernel log daemon	*Kill{SubjectIP, KlogD}*

receive the log entries for that particular event. Figure 3 shows the Probes A, B and W corresponding to the alert and the forensic profiles shown in Figure 2.

The mechanism discussed in Section 3.1 is applicable to unknown attacks. An unknown attack does not have a passive profile. However, since attacks have common events that trigger them, the alerts generated would be used to save the log entries in the forensic server that can be later used to trace the attacker and corroborate evidence. Intrusion detection systems, which are based on attack signatures, are unable to track illicit activity caused by a new or unknown attack. The forensic profiling system deals with this problem by creating an unstructured profile. If an unsolicited event does not match any of the known profiles, an unstructured profile is constructed with all the alerts generated in the same region and time span. This ensures that even if the attack was not stopped, evidence related to the activity is collected and saved.

3.3 Active Event Monitoring

Active monitoring is a process that continuously checks for variations in the intensity of events. Exponentially weighted moving average (EWMA), a statistical process control technique, is used to detect drift

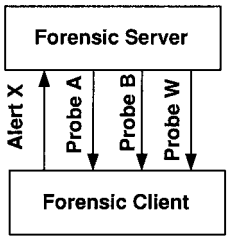


Figure 3. Sample forensic server/client communications.

in a parameter being monitored [15]. The decision regarding the state of control of a process at any time instant depends on the EWMA statistic, which is an exponentially weighted average of all the prior data and depth of memory. The EWMA control technique takes the statistic (CPU utilization, traffic intensity, log event intensity, memory utilization) that is to be monitored as argument in real time and recursively checks if the current value lies within the control limits. The control limits are determined by training data composed of usual or normal events. The testing data are interspersed with intrusive events. The events in an information system are closely related to each other. Therefore, the EWMA technique that makes use of auto-correlated data has been applied. The EWMA statistic for i-*th* observation *z(i)*, is given as:

$$z(i) = \lambda \cdot x(i) + (1 - \lambda) \cdot z(i - 1) \qquad i = 1..n \tag{1}$$

where *z(0)* is the mean of the training data, *x(i)* is i-*th* observation, n is the number of observations to be monitored and $0 < \lambda \leq 1$ is the depth of memory. The depth of memory determines the rate at which the past observations enter into the calculation of EWMA statistic. Conventionally, $\lambda = 1$ implies the most recent observation influences EWMA. Thus, larger values of λ give more weight to recent data. The depth of memory depends on the parameter being monitored, i.e., greater depth into the history of events is required for event log intensity than for CPU utilization. For our calculations, λ lies between 0.2 and 0.3 for log intensity measurement and 0.8 for CPU utilization. An anomaly is detected if *z(i)* falls outside the control limits and an alert is generated by the forensic client to the forensic server.

3.4 Active Parsing Enabling Mechanisms

Most alerts generated by the forensic clients are based on parsing log files. Thus, the efficacy of FPS depends on maximizing the logging mechanisms of forensic clients.

Continually parsing a history file helps identify the execution of suspicious commands like chattr, chmod for critical files such as log files and the history file itself. User activity logging can be configured on a Linux machine by making use of the process accounting package. All the commands used at the console are logged into a binary file. This provides more information about command execution than the history file in terms of the user who executed a command, CPU time, connection times for each user, etc. Network monitoring tools must be deployed for logging network activities. Firewalls must be configured to log connection failures, and servers must be configured to log actions specific to the services provided by the server.

4. Case Study

This section presents a case study involving a WU-FTP attack [2] on a network, interjected by the alerts and probes generated by FPS if it were part of the network. To exploit the Globbing vulnerability associated with Version 2.6.1 or earlier of a WU-FTP server, the attacker should have authenticated or anonymous access to the FTP server. Next, the attacker should be able to create a buffer overflow and then execute a process that installs a backdoor/Trojan/rootkit. The following subsections describe the victim network and the trail of events associated with the attack.

4.1 Victim Network

The internal network of the company and its DMZ setup is well-designed from a security perspective. The DMZ consists of the standard set of network servers (web, email, DNS servers and a dedicated FTP server for distributing hardware drivers for the company inventory). Two firewalls are used, one separating the DMZ from the Internet and the other separating the DMZ from the internal network (LAN). Two network IDSs are part of the DMZ setup. Individual firewalls are installed in each of the DMZ machines. No connections are allowed from the DMZ to either the Internet or to the LAN. Also, no connections are allowed between the DMZ machines themselves. An outside machine may only connect to a single port of each DMZ host. Each DMZ machine runs a host-based firewall.

The forensic server is an integral part of FPS as it maintains the forensic profile database. Since the focus is on the WU-FTP attack, its forensic profile descriptor is shown in Figure 4.

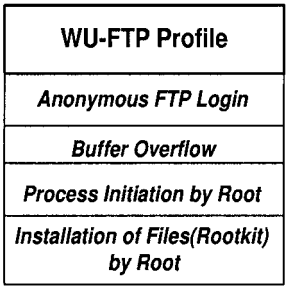


Figure 4. Forensic profile for the WU-FTP attack.

All the network components are configured as forensic clients by installing agents. Unauthorized network connections are detected by firewalls, which issue alerts in the following format:

$$FailCon\{SourceIP{:}Port, DestinationIP{:}Port, AttemptNo\}$$

4.2 Attack Progression

The forensic team was notified after a customer was unable to connect to the company's FTP server. It was subsequently discovered that the server's operating system had been deleted. Live forensic analysis of the machine could not be performed as the server had crashed while it was unattended. The log files of the FTP server also could not be recovered as the syslog was not configured for remote logging. Therefore, it was only possible to analyze the hard drive of the FTP server and logs retrieved from the IDS and firewalls. Analyzing the 20GB hard drive for forensic evidence was deemed to be too time consuming.

The forensic investigation revealed that the IDS detected the WU-FTP attack on Apr 1 02:29:45. The FPS response mechanism prompted the IDS to send the following alert to the forensic server.

- Alert generated by IDS to the forensic server:

When	Subject	Object	Action
Apr 1 02:29:45	IP Addr IDS	IP addr FTP server	WU FTPD attack

The forensic server reacted to the alert by issuing probes. Alerts can be generated by two or more forensic clients simultaneously depending on the type of event detected by the clients. Suspicious activity detected by a forensic client will not, on its own, determine the progression of an attack. But an IDS can detect an attack as a whole. Thus, the implementation of the forensic profiling system differs in the steps followed in tracking the attack depending on the client that triggers the alert. (The

alerts to the server may be sent in a random fashion.) We examine the flow of events when the attack was detected by the IDS.

Check probes were generated simultaneously because, although they are related from the point of view of the attack and occur in a chronological order, they are independent of each other. The probes generated by the forensic server corresponded to the alerts contained in the descriptor for the WU-FTP attack. Therefore, the Check_Probes sent were:
(i) Check_Probe sent to the FTP server:

Dest Addr	CheckFlag	Time
IP Addr FTPserver	FTPLogin	Apr 1 02:29:45

If the FTP server had sent a NULL packet, this would have indicated that no one was logged into the FTP server at the time. Otherwise, the FTP server would have responded by providing the IP addresses that were logged in at the time. The server went over the logs it captured through the forensic clients and scanned for a match for the IP addresses sent by the FTP server. A matched IP address is suspicious because the forensic server has logs only for suspicious activities. The forensic server then sent GetLog_Probe to the FTP server which specified the suspicious IP address as the keyword. The following alert showed that 192.1.2.3 is a suspicious IP address.

Dest Addr	GetLogFlag	Keyword	Time
IP addr FTP Server	Set	192.1.2.3	Apr 1 02:29:45

The log fragments that corroborated the alert recovered from the FTP server are presented below.

FTP System Logs:

Apr 1 00:08:25 ftp ftpd[27651]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], mozilla@
Apr 1 00:17:19 ftp ftpd[27649]: lost connection to 192.1.2.3 [192.1.2.3]
Apr 1 00:17:19 ftp ftpd[27649]: FTP session closed
Apr 1 02:21:57 ftp ftpd[27703]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], mozilla@
Apr 1 02:26:13 ftp ftpd[27722]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], mozilla@
Apr 1 02:29:45 ftp ftpd[27731]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], x@

(ii) Check_Probe sent to the IDS:

Dest Addr	CheckFlag	Time
IP Addr IDS	Buffer Overflow	Apr 1 02:29:45

If the response sent by the IDS to the server contained a NULL packet, a buffer overflow condition would have been negated. Otherwise the forensic server would have sent the GetLog_Probe.

(iii) Check_Probe sent to the FTP server:

Dest Addr	CheckFlag	Time
IP Addr IDS	Process Execution + Root Access	Apr 1 02:29:45

If the response sent by the FTP server to the forensic server contained a NULL packet, this would have implied that no script was running on the server. Otherwise, the forensic server would have sent the GetLog_Probe. The forensic server would have continuously sent GetLog_Probes to the FTP server. If the FTP server had crashed as a result of the attack, it would have stopped serving the GetLog request initiated by the forensic server.

After gaining access to the FTP server, the attacker tried to connect to his machine (192.1.2.3), which was not allowed. Also, the attacker attempted to connect to the mail server. This is implied by the following FTP connection logs.

FTP Connection Logs:

Apr 1 02:30:04 ftp ftpd[27731]: Can't connect to a mailserver.
Apr 1 02:30:07 ftp ftpd[27731]: FTP session closed

The corresponding alert indicated that an unauthorized network connection attempt was generated by the FTP server.

- Alert generated by FTP server to the forensic server for connection failure to mail server:

When	Subject	Object	Action
Apr 1 02:30:04	IP Addr FTP Server	IP Addr Mail server	FailCon

A similar alert was generated by the firewall.

The attacker was able to gain root access, upload a file and later execute a script. This can be inferred from the FTP transfer logs.

FTP Transfer Logs:

Mon Apr 1 02:30:04 2002 2 192.1.2.3 262924 /ftpdata/incoming
/mount.tar.gz b _ i a x@ ftp 0 * c

The alert generated by the FTP server to the forensic server, indicative of uploading a file with root privileges, is shown below.

- Alert generated by the FTP server to the forensic server indicating file upload:

When	Subject	Object	Action
Mon Apr 1 02:30:04 2002	IP Addr FTP Server	mount.tar.gz	FileUpload

The attacker was able to upload files on the FTP server as the FTP server was world writable. This file is the suspected rootkit. The attacker later deleted the operating system causing the FTP server to crash.

5. Conclusions

This paper proposes a forensic profiling system (FPS) for real-time forensic evidence collection. A dedicated forensic server maintains an audit trail embedded in a forensic profile. Because FPS keeps track of anomalous activities in a network, the time spent on filtering system log files during a forensic investigation is drastically reduced. FPS also makes it easier to retrieve the logs of crashed hosts as the hosts can send log entries associated with alerts to the forensic server. Since all attacks have a general commonality, unknown attacks can be tracked by the forensic sever on the basis of the alerts generated by forensic clients. A detailed investigation of attacks is required to construct forensic profiles. Also, it is necessary to evaluate the overhead involved in active parsing and monitoring.

References

[1] J. Barrus and N. Rowe, A distributed autonomous agent network intrusion detection and response system, *Proceedings of the Command and Control Research Technology Symposium*, pp. 577-586, 1998.

[2] A. Chuvakin, FTP Attack Case Study, Part I: The Analysis (www.linuxsecurity.com/feature_stories/ftp-analysis-part1.html) 2002.

[3] F. Cuppens, Managing alerts in a multi intrusion detection environment, *Proceedings of the Seventeenth Annual Computer Security Applications Conference*, 2001.

[4] F. Cuppens and A. Miège, Alert correlation in a cooperative intrusion detection framework, *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[5] H. Debar and A. Wespi, Aggregation and correlation of intrusion detection alerts, *Proceedings of the Fourth International Workshop on Recent Advances in Intrusion Detection*, pp. 85-103, 2001.

[6] M. Huang, R. Jasper and T. Wicks, A large-scale distributed intrusion detection framework based on attack strategy analysis, *Proceedings of First International Workshop on Recent Advances in Intrusion Detection*, 1998.

[7] C. Kahn, D. Bolinger and D. Schnackenberg, Common Intrusion Detection Framework (www.isi.edu/gost/cidf/), 1998.

[8] P. Ning, Y. Cui and D. Reeves, Constructing attack scenarios through correlation of intrusion alerts, *Proceedings of the Ninth ACM Conference on Computer Security*, 2002.

[9] P. Ning, X. Wang and S. Jajodia, A query facility for the common intrusion detection framework, *Proceedings of the Twenty-Third National Information Systems Security Conference*, pp. 317-328, 2000.

[10] P. Ning, X. Wang and S. Jajodia, Abstraction-based intrusion detection in distributed environments, *ACM Transactions on Information and System Security*, vol. 4(4), pp. 407-452, 2001.

[11] P. Porras and P. Neumann, EMERALD: Event monitoring enabling responses to anomalous live disturbances, *Proceedings of the Twentieth National Information Systems Security Conference*, pp. 353-365, 1997.

[12] K. Shanmugasundaram, N. Memon, A. Savant and H. Bronnimann, ForNet: A distributed forensics network, *Proceedings of the Second International Workshop on Mathematical Methods, Models and Architectures for Computer Network Security*, 2003.

[13] A. Valdes and K. Skinner, Probabilistic alert correlation, *Proceedings of the Fourth International Workshop on the Recent Advances in Intrusion Detection*, 2001.

[14] J. Yang, P. Ning and X. Wang, CARDS: A distributed system for detecting coordinated attacks, *Proceedings of the IFIP TC11 Sixteenth Annual Working Conference on Information Security*, 2000.

[15] N. Ye, S. Vilbert and Q. Chen, Computer intrusion detection through EWMA for autocorrelated and uncorrelated data, *IEEE Transactions on Reliability*, vol. 52(1), pp. 75-81, 2003.

Chapter 14

GLOBAL INTERNET ROUTING FORENSICS

Validation of BGP Paths using ICMP Traceback

Eunjong Kim, Dan Massey and Indrajit Ray

Abstract The Border Gateway Protocol (BGP), the Internet's global routing protocol, lacks basic authentication and monitoring functionality. Thus, false routing information can be introduced into the Internet, which can cause the total collapse of packet forwarding and lead to denial-of-service or misdirected traffic. While it may be impossible to prevent such an attack, we seek to provide the routing infrastructure with a mechanism for identifying false paths through efficient validation, proper recording and forensic analysis of routing data. Towards this end, we propose a novel BGP path verification technique using ICMP traceback messages that has been extended to include AS-PATH and link connectivity information. The approach can be easily deployed as it does not require modifications to BGP.

Keywords: Routing forensics, BGP, ICMP traceback

1. Introduction

The Internet plays an increasingly important role in commerce, government and personal communication. A large-scale attack (or even an unintended operational error) can seriously disrupt service to critical sectors and have a major impact on the economy. In response, a variety of end system security techniques, such as encrypted connections and VPNs have been proposed. However, almost all of these systems rely on the unsecured Internet infrastructure to compute routes and deliver packets. If the Internet infrastructure fails to deliver data packets, there is very little the end systems can do to recover. This paper examines techniques for detecting invalid routes in the Internet infrastructure and

presents an effective approach for gathering and extracting routing data from the network that can be used for forensic analysis.

At the global infrastructure level, the Internet consists of thousands of Autonomous Systems (ASs), each identified by a unique number. An AS can be viewed as a group of links and routers that are under the same administrative control. The ASs are responsible for routing information over the Internet backbone. The Border Gateway Protocol (BGP) [5] is the de facto inter-AS routing protocol; it is used to exchange reachability information between ASs. BGP is designed to cope with events that alter the structure of the Internet, such as the addition of new links and new ASs, the failure (temporary or long lasting) of links, and changes in routing policies. However, BGP contains very limited security mechanisms and thus presents several interesting challenges for path validation and routing forensics.

BGP implicitly assumes that routers advertise valid information. For example, suppose that AS 12145 (Colorado State University) incorrectly (maliciously) reports that it has a direct connection to `www.largecompany.com`. Other BGP routers will believe this route and portions of the Internet will select this path as the best route to `www.largecompany.com`. When the traffic arrives at AS 12145, the traffic may simply be dropped or someone may attempt to spoof the `www.largecompany.com` website. As a result, `www.largecompany.com` may notice a drop in traffic. If AS 12145 later withdraws its false route, BGP routers at some point will simply switch back to the valid path. However, it will take a very long time for the changes to propagate throughout the Internet. In addition, owing to the large number of BGP destinations and the large volume of BGP routing changes, a particular BGP path change is unlikely to trigger any alarms at remote sites. Nonetheless, such actions have the potential to significantly disrupt the affected site. Extracting enough routing information from the network so as to be able to identify the reason for this lost traffic (namely, that it has been triggered by some AS announcing an invalid path information) is quite challenging with current techniques.

This paper presents an approach for monitoring, gathering and validating a route to a destination. The technique works as follows. Suppose AS_1 has incorrect path information for AS_2. This can be due to one of several reasons, e.g., malicious advertisement of wrong path information by a neighboring AS of AS_1 or misconfiguration at AS_1. Under our approach, AS_2 will eventually know that AS_1 has an incorrect path information about AS_2.[1] In addition, AS_2 has the potential to know what other ASs have invalid path information about it. If AS_1 (and the

other ASs) are reachable from AS_2, then AS_2 can alert these ASs that incorrect path information has been introduced.

The proposed approach uses ICMP (Internet Control Message Protocol) traceback messages. As data packets flow through routers, occasional packets (one in twenty thousand) generate ICMP traceback messages. These traceback messages allow a destination to reconstruct the path used to reach the destination. Unlike other approaches that attempt to monitor or validate all paths, our methodology focuses on paths that actively carry data traffic. There may be more than 18,000 ASs that have some path to `www.largecompany.com`, but relatively few of these sites may be actively sending data traffic. By using the ICMP traceback mechanism, monitoring and validation messages are only sent for paths that are actively in use. The ICMP traceback messages are enhanced with AS-PATH information and link connectivity information. Also, traceback messages are sent along multiple (ideally disjoint) paths to reduce the probability that packets are (maliciously or otherwise) dropped or corrupted. Thus, a router can dynamically keep track of paths used to reach the destination, monitor routing changes for the actively used paths to this destination, and produce logs that can be used to reconstruct routes in the event of a suspected attack. As a side-effect, this approach provides a more fault-tolerant, fault-resilient, reliable and secure BGP routing for the Internet infrastructure.

2. Enhanced BGP iTrace

In the original ICMP traceback proposal [1], ICMP traceback (iTrace) is defined to carry information on routes that an IP packet has taken. This mechanism is used to deal with denial-of-service attacks by verifying the source IP address. When an IP packet passes through a router, iTrace is generated with a low probability of about 1/20,000 and sent to the destination. Lee, *et al.* [2] propose using cumulative IP information to verify the true IP packet origin. When a router receives a IP packet and forwards it, it generates an iTrace message and appends its own IP address; this iTrace message is sent to the next hop instead of to the destination. When a router receives an iTrace message, it appends its own IP address to the iTrace message. Mankin, *et al.* [4] have proposed an "intension-driven" version of iTrace. However, at best, their messages simply record the path of links and routers that packets may have taken. They provide no information on why a router selected a particular next hop. To provide reliable and fault-tolerant BGP routing protocol, it is necessary to add appropriate mechanisms for monitoring and authenticating paths. BGP is a policy-based routing protocol and

each AS chooses the path among the multiple routes it receives from its neighbors for the same prefix according to its own criteria. An AS also can apply a policy when exporting a route. Generally, ASs filter incoming or outgoing announcements to implement policies such as peering and transit. Filtering can be implemented using prefix filters, access lists and route maps. Using these basic primitives and a few others, an AS can control the flow of announcements between its routers and their BGP peers [3]. Our approach uses advanced filtering and ICMP traceback to provide both path and origin validation. However, adding more functionality into routers is not recommended as routers already handle many complicated functions. Therefore, our approach requires a separate server or a process that provide security mechanisms.

2.1 Modified ICMP Traceback Messages

Our approach uses an extended form of the ICMP traceback (iTrace) message. Instead of authenticating BGP announcement messages and updating messages, it uses the actual data traffic to collect proper connectivity information for AS-PATH and prefix origin validation. As data packets traverse a route, each router on the path generates iTrace messages. These iTrace messages contain information about the traced packet source and destination address, previous link, and the AS-PATH which each router finds in its routing table to reach the destination.

Table 1 presents the list of tags for message elements. We add the last three tags, *0x10* for *Traced Packet Source Address*, *0x11* for *Traced Packet Destination Address*, and *0x12* for *AS-PATH* information. The other elements in Table 1 are defined in [1]. In the following, we briefly discuss the three new tags.

Traced Packet Source Address (TAG = 0x10)/Traced Packet Destination Address (TAG = 0x11): This element contains the traced packet source address/destination address, which is 4 octets for an IPv4 address and 6 octets for an IPv6 address; hence, the LENGTH field is either 0x0004 or 0x0006. The element format is presented in Figure 1.

AS-PATH Information (TAG = 0x12): This element contains AS-PATH information, which is found in a BGP routing table. The length of the element is variable since the number of ASs on the path is not fixed. The element format is almost the same as in Figure 1 except for the LENGTH(variable) and VALUE(variable length) fields. The Back Link element is used for link connectivity information from the perspective of

Table 1. ICMP traceback tags [1].

Tag	Element Name
0x01	Back Link
0x02	Forward Link
0x03	Interface Name
0x04	IPv4 Address Pair
0x05	IPv6 Address Pair
0x06	MAC Address Pair
0x07	Operator-Defined Link Identifier
0x08	Timestamp
0x09	Traced Packet Content
0x0A	Probability
0x0B	RouterId
0x0C	HMAC Authentication Data
0x0D	Key Discloser List
0x0E	Key Discloser
0x0F	Public-Key Information
0x10	*Traced Packet Source Address*
0x11	*Traced Packet Destination Address*
0x12	*AS-PATH Information*

the iTrace message generator. In the VALUE field, an AS number pair is added for one of the sub elements.

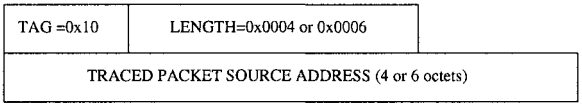


Figure 1. Traced packet source address element format.

2.2 AS-PATH Validation

Figure 2 shows how the approach works for path validation. In the example, the CSU web server (129.82.100.64) is connected to AS1. The AS-PATH from UCLA (131.179.96.130) to the CSU web server is [AS8 AS7 AS6 AS1]. When the UCLA client sends data to the CSU web server, the data traffic traverses this path (solid line with arrows). When a data packet is sent by a client from a UCLA machine, all the routers along the path (AS8, AS7, AS6, AS1) generate iTrace messages with a probability of 1/20,000. When the data packet traverses the AS7 router, it generates iTrace messages with the data packet's source address (131.179.96.130) and the data packet's destination address (129.82.100.64), its previous

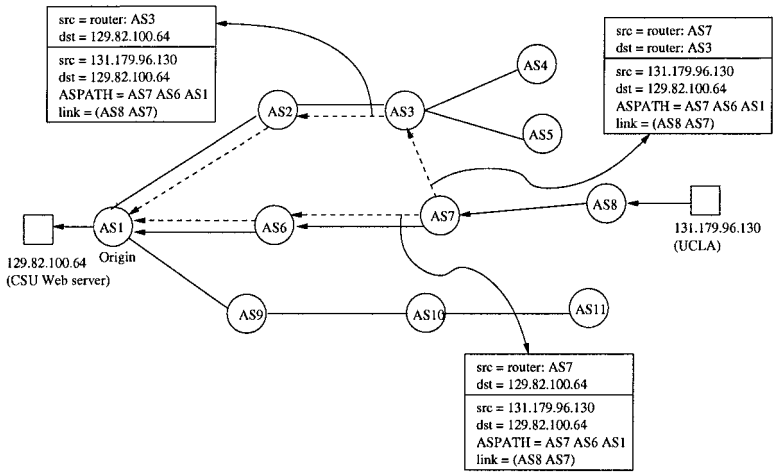


Figure 2. Valid path verification with ICMP traceback messages.

link as (AS8 AS7) and the AS-PATH from itself to the destination [AS7 AS6 AS1]. This AS-PATH is found in AS7's BGP routing table. When router AS7 forwards the data packet, it generates two identical iTrace messages. One iTrace message is attached to the ICMP header, which has AS7 as its source and the same destination as the data packet's destination (129.82.100.64). The other iTrace message is attached to the ICMP header which has AS7 as its source but a destination as an arbitrary node (AS3 in example), which hopefully has different path to reach the destination. When AS3 receives an iTrace message, it simply changes the ICMP header to send the iTrace message to the data packet's destination. The new ICMP header has AS3 as its source and 129.82.100.64 as its destination. We do not discuss how a node is picked to send the iTrace message as it is outside the scope of this paper. Instead, we simply assume that a random node is selected by the router; the only restriction is that the node should know how to handle iTrace messages. Other intermediate AS routers operate similarly to AS7 when they propagate data packets to their destinations. However, the iTrace messages generated by each router have slightly different information. One iTrace message, which is received by AS3, traverses along the path, [AS3 AS2 AS1], to reach the destination. The other iTrace message, which is directly sent to the destination, follows the path, [AS7 AS6 AS1]. When the data packet arrives in AS6, the router follows the same procedure as AS7 to generate and send iTrace messages. All the other routers (AS4, AS5, AS9, AS10, AS11) do not see the data packets and iTrace messages.

At the destination, the router first checks each iTrace message's source field. It finds three different iTrace messages with the same source, 131.179.96.130. One is generated by AS6, another is generated by AS7 and the third is generated by AS8. The router constructs the path from the source to the destination based on link information: Link() (this means client is directly connected), Link(AS8 AS7) and Link(AS7 AS6), and path information: [AS8 AS7 AS6 AS1], [AS7 AS6 AS1] and [AS6 AS1]. If there are no AS-PATH conflicts, the router regards AS-PATH, [AS8 AS7 AS6 AS1], as a valid path from UCLA (131.179.96.130) to the CSU web server (129.82.100.64).

The destination router constructs a path tree or a path set for all source and destination pairs. If the destination uses a path tree, the router builds a path tree from the information, which is collected by all the iTrace messages it receives. The path tree has itself as the root node; its leaves correspond to the source addresses of data packets. Each path on the tree from the root to a leaf corresponds to an AS-PATH. If the destination uses a path set, a collection of paths is created from all sources. The decision between constructing all paths from sources to this node and building one path tree is an implementation issue that depends on efficiency, space overhead and performance.

When a destination node receives an iTrace message, it compares the new information with previous information. Any inconsistency triggers an alarm. Three different situations can exist, and the reaction of the destination to each is different. The first is when AS-PATH is not directly connected to the destination, e.g., destination node, AS3, gets an iTrace message with AS-PATH: [AS1 AS2 AS3] and AS2 is not its next hop neighbor. This is an obvious sign of attack; therefore, the router immediately sets a flag and sends an emergency message to the system operator. The second situation is when AS-PATH is not consistent, i.e., it does not match any previous AS-PATH information. This can be interpreted in two possible ways: one is an attack in which a false origin or malicious router sends wrong reachability information to its neighbors, and the other is misconfiguration. However, we do not distinguish misconfiguration from an attack since the effects are same. The third situation occurs when one router on the path announces wrong AS-PATH information to make the AS-PATH longer than the real one. This occurs when a router misconfigures the path to reach the destination or intentionally injects wrong reachability information. In this case, our approach detects the false AS-PATH based on missing path derivation. Because real data traffic does not traverse routers which are not on the path, the destination never receives iTrace messages from them.

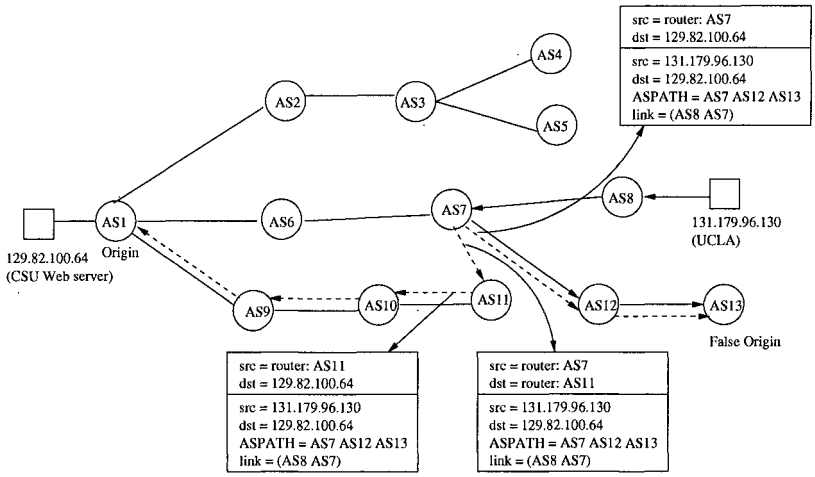


Figure 3. Invalid path with false origin.

In the following, we present examples of the scenarios and demonstrate how they can be detected via AS-PATH validation with iTrace.

2.3 BGP iTrace Under Attacks

Figure 3 presents a possible attack scenario. AS13 is a false origin which impersonates as the owner of the CSU web server. In this case, the AS-PATH to reach the destination 129.82.100.64 is [AS8 AS7 AS12 AS13]. The data traffic from UCLA (131.179.96.130) uses this false path. Even though the correct path in this example is [AS1, AS6, AS7, AS8], the intermediate routers on the false path simply propagate all the data packets sent from UCLA to the wrong destination. This is because these intermediate routers cannot see the entire network topology. All these routers generate iTrace messages with the wrong path information. AS7, in particular, generates two iTrace messages with the wrong AS-PATH – [AS7 AS12 AS13]. One of these iTrace messages is sent to the false destination AS13, and the other to the neighboring node AS11. AS11 forwards this iTrace message to the correct destination, which then detects a path inconsistency. It is quite possible that AS11 sends an iTrace message to the false destination. However, because of the rich connectivity of the Internet, there is high probability that an iTrace message is sent to a node that has the path to the correct destination. Indeed, if an iTrace message is sent as far as possible from the iTrace generator, the message has a good chance of reaching the correct destination.

When the iTrace message reaches the correct destination, the router notes that the AS-PATH is [AS7 AS12 AS13], which is generated by

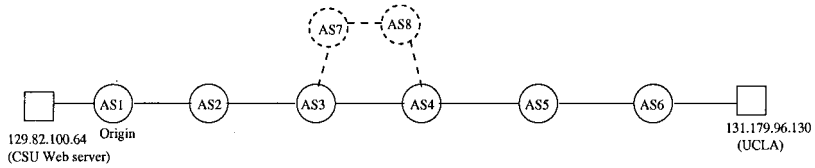


Figure 4. Invalid path with false reachability information.

Table 2. AS-PATH information collected by the the destination.

iTrace Originator	AS-PATH
AS2	[AS2 AS1]
AS3	[AS3 AS2 AS1]
AS4	[AS4 AS8 AS7 AS2 AS1]
AS5	[AS5 AS4 AS8 AS7 AS2 AS1]

AS7. The router recognizes that this information is incorrect as AS13 does not match the anticipated value (AS8), and AS12 is not its NEXT-HOP. This is an obvious attack; a flag is set and a report is sent to the system operator. Thus, with this iTrace message, the destination node is not only able to verify the incorrect AS-PATH, but also detect and locate the false origin.

Figure 4 presents another example. Here, AS-PATH from UCLA to CSU is [AS6 AS5 AS4 AS3 AS2 AS1]. Somehow, AS4 reflects that the AS-PATH to reach CSU web server is [AS4 AS8 AS7 AS3 AS2 AS1]. Based on this reachability information, AS5 has the AS-PATH to reach the same destination as [AS5 AS4 AS8 AS7 AS3 AS2 AS1]. When data packets are sent from UCLA, all the routers along the path generate iTrace messages. AS1 collects and examines each of these iTrace messages. The resulting accumulated AS-PATH information at AS1 is shown in Table 2. No inconsistencies are noted, but the destination never gets iTrace messages originating from AS7 or AS8. After a sufficiently long time, if the destination does not receive any direct AS-PATH information from both AS7 and AS8, the destination will suspect that neither AS7 nor AS8 are on the path that data packets traverse. The plausible causes at this stage are either that AS4 obtains incorrect reachability information from its neighbors or that AS4 injects this information itself. Based solely on the AS-PATH information, the cause cannot be precisely determined. In this case, the destination triggers an alarm and notifies the operator of this observation. Further analysis is required at this stage to diagnose the problem.

AS-PATH and AS Origin Validation Algorithm: Our AS-PATH validation approach differs from techniques that authenticate AS-PATH information in BGP routing announcement or update messages. These techniques need an additional mechanism to validate the prefix origin. This is because AS-PATH validation, by itself, does not guarantee the authentication of prefix origin. In our approach, the destination router independently derives AS-PATH from iTrace messages based on real traffic. Indeed, AS-PATH information from iTrace messages provides partial or complete views of a path from source to destination. Since a prefix origin corresponds to the last router of AS-PATH, our approach does not require a separate validation process.

ALGORITHM 1 *AS-PATH Validation Algorithm*
Input: *iTrace messages*
Output: *report message*

```
Procedure ASPathValidation
begin
    /* longest(s, d) is longest AS-PATH from source
       (s) to destination (d),
       longestSet is a collection of longest(s, d) */
  longest(s, d) = null; longestSet = {}; tracedAS = {}
  timer = 5min
   while forever do
     switch (event)
     event an iTrace message has arrived do
        begin
          remove the ICMP header
          get (s, d) source and destination of iTrace message
          get ASPATH from iTrace message
          get sendAS from iTrace message
          get longest(s, d) from longestSet
         /* Check if AS-PATH is directly connected with itself */
          if the last link of ASPATH ≠ NEXTHOP
            /* this is an attack */
                set a flag and send an emergency message to the operator
          else
              if ASPATH is subpath of longest(s, d)
                  tracedAS = tracedAS ∪ sendAS
                  /* current longest path is shorter than ASPATH */
              else if longest(s, d) is subpath of ASPATH
                      longestSet = longestSet − longest(s, d)
                      longest(s, d) = ASPATH
                      tracedAS = tracedAS ∪ sendAS
                      longestSet = longestSet ∪ longest(s, d)
              else
                     /* AS-PATH is inconsistent */
                     send inconsistent path warning message to operator
```

```
            endif
        endif
      end
    event timer is expired do
      begin
        /* there are some subpaths which are never received */
        if ∃ AS ∈ longest(s, d) and AS ∉ tracedAS
            send unreceived subpath warning message to operator
            longest(s, d) = null; longestSet = {}
            tracedAS = {}
            set timer with 5 minutes
        endif
      end
  endwhile
end
```

3. Conclusions

This paper describes a technique for fortifying the Internet routing infrastructure with a mechanism to identify false path information. The approach, based on efficient validation, proper recording and forensic analysis of routing data, integrates several partial solutions that have been proposed elsewhere. The ICMP traceback (iTrace) is adapted to provide efficient path validation mechanisms. In particular, the iTrace message is modified to include important BGP information such as Source AS, link connectivity information and AS-PATH information. The iTrace message facilitates checking the validity of paths. A unique feature is that real traffic is used to validate paths. Furthermore, filtering, local database management, path and origin verification work in a fully distributed manner and guarantee good availability and scalability.

It is important to note that the proposed approach does not use cryptographic techniques. This is because public key schemes require an established PKI that involves significant overhead to generate and verify signatures; this affects scalability and deployability using the existing infrastructure. In contrast, our approach depends on the distributed nature of the Internet to spread the correct information and corroborate paths, and it uses the Internet topology to detect impersonated routes and invalid paths.

Recent studies have shown that implementation and/or misconfiguration errors are responsible for a significant portion of traffic [3]. However, in this work, we do not take any extra steps to differentiate between these errors and malicious attacks because both cause the same reachability and convergence problems.

The proposed approach provides security mechanisms without any operational degradation of BGP. Also, it facilitates incremental deployability and scalability that adapt well to the real world.

Notes

1. Currently, the same information can be obtained by a BGP administrator going over BGP log records which can be in the millions. However, no mechanism exists that will alert the BGP administrator to go over the log records.

References

[1] S. Bellovin, ICMP traceback messages, Internet Draft, March 2001.

[2] H. Lee, V. Thing, Y. Xu and M. Ma, ICMP traceback with cumulative path: An efficient solution for IP traceback, *Proceedings of the Fifth International Conference on Information and Communications Security*, pp. 124-135, 2003.

[3] R. Mahajan, D. Wetherall and T. Anderson, Understanding BGP misconfiguration, *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pp. 3-16, 2002.

[4] A. Mankin, D. Massey, C. Wu and L. Zhang, On design and evaluation of intention-driven ICMP traceback, *Proceedings of the Tenth IEEE International Conference on Computer Communications and Networks*, pp. 159-165, 2001.

[5] Y. Rekhter and T. Li, Border Gateway Protocol 4, RFC 1771, July 1995.

[6] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. Wu and L. Zhang, Protecting BGP routes to top level DNS server, *IEEE Transactions on Parallel and Distributed Systems*, vol. 14(9), pp. 851-860, 2003.

Chapter 15

USING SIGNALING INFORMATION IN TELECOM NETWORK FORENSICS

T. Moore, A. Meehan, G. Manes and S. Shenoi

Abstract Telephones are often used to facilitate criminal and terrorist acts. The signaling core of public telephone networks generates valuable data about phone calls and calling patterns, which may be used in criminal investigations. However, much of this data is not maintained by service providers and is, therefore, unavailable to law enforcement agencies. This paper presents techniques for collecting and storing important data pertaining to phone calls and calling patterns. The techniques leverage existing telecommunications network resources, and address the long-term storage issues of massive amounts of call record data.

Keywords: Telecommunications networks, signaling messages, call detail records

1. Introduction

Detailed information about telephone calls and the calling patterns of suspects can be very useful in criminal and terrorism investigations. But the call detail records currently collected by service providers for billing purposes are not comprehensive enough, nor are they retained for more than a few months.

A modern telecommunications network incorporates a transport network that carries voice and data, and a vital signaling network core that controls and manages voice and data circuits in the transport network. Call setup and other messages that traverse the signaling network are routinely examined by service providers to maintain quality of service, debug network problems and generate billing records. These messages provide a wealth of forensic information about phone calls and calling patterns. Since signaling messages provide data about phone calls – not the content of phone conversations – collecting and analyzing these

messages may not be subject to the same legal restrictions as recording voice conversations.

This paper describes techniques for collecting detailed information about phone calls and calling patterns. The techniques can be implemented using current telecommunications network survellance equipment (e.g., [9]), and the collected data can be analyzed and stored at little additional cost.

The following sections describe signaling networks and techniques for collecting signaling data pertaining to telephone calls and calling patterns. The storage requirements for the techniques are analyzed, and a post-capture data processing technique that addresses the long-term storage issues of massive amounts of call record data is proposed.

2. Signaling Networks

Public telephone networks incorporate a transport network that carries voice and data, and a vital (out-of-band) signaling network that controls voice and data circuits in the transport network. The Signaling System 7 (SS7) protocol and its variations are used worldwide in signaling networks [7, 10, 11]. SS7 is responsible for setting up calls and implementing advanced features such as calling cards and toll-free service. VoIP and wireless networks use different protocols, but interface to public telephone networks using SS7. The following subsections describe the SS7 network architecture, SS7 messages that provide forensic information, and strategies for collecting SS7-related call data.

2.1 SS7 Overview

SS7 networks have three types of components (signaling points): service switching points (SSPs), signal transfer points (STPs) and service control points (SCPs) (see Figure 1). Each signaling point has a unique point code for routing SS7 messages. SSPs are central office switches that connect phones to voice trunks (multiple solid lines in Figure 1); they also generate SS7 messages for call setup and database queries.

STPs receive and route SS7 messages between signaling points using out-of-band signaling links (dashed lines in Figure 1). In U.S. telephone networks, each SSP is connected to at least one mated pair of STPs [12, 13]. SCPs (not shown in Figure 1) mainly provide database access for advanced services, e.g., call forwarding and toll-free numbers; like SSPs, they connect to STPs via signaling links.

SS7 messages contain important control data about telephone calls, e.g., calling and called party numbers, as well as the time and duration of calls. Collecting this data does not require an inordinate amount

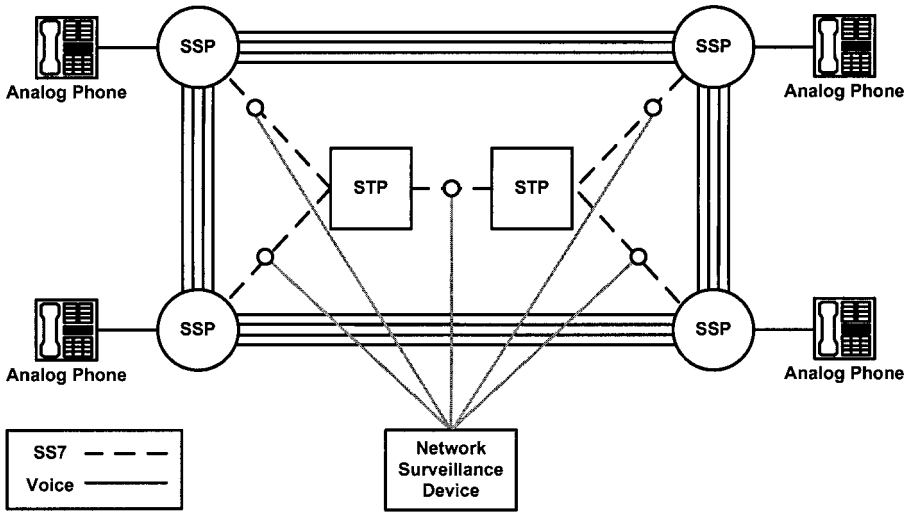


Figure 1. Signaling System 7 network.

of resources – many U.S. providers already employ surveillance devices [8, 9] to monitor SS7 links (see Figure 1). In U.S. networks, these devices are co-located at STPs as all signaling links pass through STPs. Some countries (e.g., U.K.) employ "fully-associated" networks, where SSPs are connected directly by signaling links (not via STPs as in U.S. networks). Such a topology makes it impractical to monitor every SS7 link.

2.2 SS7 Messages

SS7 messages contain valuable forensic information. Call setup is governed by the ISUP protocol [1]. One of the most important ISUP messages is the initial address message (IAM), which initiates phone calls. Other messages for setting up calls – address complete (ACM), answer (ANM), release (REL) and release complete (RLC) messages – also contain useful data. Some ISUP messages, e.g., block (BLO) messages that remove voice circuits for maintenance, are very powerful but are used rarely.

Database queries and responses are implemented by TCAP messages [4]. These general-purpose messages translate toll-free and ported phone numbers. Note that the formats of the message payloads vary considerably, so any analysis software must be tailored to specific applications. The underlying SCCP header in TCAP protocol messages, however, does provide useful routing and service information [3].

MTP messages used for network management provide data about network behavior and response [2]. They include transfer prohibited (TFP) messages sent by STPs during service outages, and transfer controlled (TFC) messages used to deal with network congestion.

Emergency communications systems – 911 and GETS in the United States – use the SS7 infrastructure. When a 911 call is placed, SS7 messages are sent to a dedicated 911 tandem switch to set up the voice path. From the signaling point of view, the only difference is that 911 call setup messages have higher MTP priority values than those for normal calls. Storing SS7 message traffic destined for a 911 tandem is a simple and effective way to record activity, especially for *post mortem* analysis in the event of accidental service outages or outright attacks.

2.3 Message Collection Strategies

SS7 networks generate much less traffic than IP networks; even so, discretion must be applied when capturing traffic for forensic purposes. We consider three strategies: collecting all messages, collecting messages based on protocol type, and collecting messages based on message type.

The simplest approach is to capture all SS7 message traffic. SS7 links have low bandwidth (56 Kbps); nevertheless, this strategy is too simplistic. First, large providers may operate tens of thousands of links, making the storage of all SS7 traffic prohibitive. Second, most SS7 messages are irrelevant for forensic purposes. For example, large numbers of special fill-in signal units (FISUs) are transmitted to maintain proper timing.

A better approach is to capture traffic for specific protocols, e.g., ISUP and MTP. This is achieved by filtering messages based on their service indicator octet (SIO) field. This approach is simple and computationally efficient. The vast majority of ISUP traffic deals with call setup, so little extraneous information is stored when generating call records. Moreover, most network surveillance devices can capture protocol-specific traffic [8, 9].

Since not all messages contain useful forensic information, it is prudent to capture only specific types of messages. The corresponding message capture algorithm must examine the SIO to verify the protocol and the first byte after the routing label to determine the message type. Only messages whose protocols and types are of interest are collected.

Many network surveillance systems support network-wide message correlation and storage for *post mortem* analysis [8, 9]. However, they filter traffic based on protocol type, not message type. Nevertheless, adding this capability is straightforward. While the storage requirements are reduced, more processing is needed to filter messages in real

Table 1. Basic call record data from call setup messages.

Call Record Data	ISUP Message
Incoming Calls	IAM, ANM
Outgoing Calls	IAM, ANM, REL
Call Duration (Hold Time)	IAM, ANM
Call Duration (Conversation Time)	IAM, ANM, REL

time. Applications that require only one or two specific types of messages to be collected would benefit from this strategy. On the other hand, applications that collect practically all the traffic for a given protocol, e.g., ISUP-based call record data for forensic investigations, need not use this approach.

3. Call Record Signatures

Law enforcement agencies are primarily interested in calls involving specific phone numbers. This information can be obtained from three ISUP call setup messages: IAMs, ANMs and RELs (see Table 1). Note that SS7 is only used to connect voice circuits of different switches (SSPs). Therefore, no SS7 messages are generated for a call to a number serviced by the same switch. Still, the benefits of obtaining historical data, even if only for inter-switch calls, are quite significant. See [1, 7] for details of the call setup process.

Initial address messages (IAMs) contain three key parameters: called party number, calling party number and circuit identification code (CIC), which indicates the voice circuit used for call setup. Only IAMs contain the called and calling party numbers; other call setup messages include just the CIC. The CICs in these other messages must match with the IAM's CIC for the messages to be correlated with the IAM.

The original called number parameter in an IAM also conveys useful information. When a ported or toll-free number is dialed, the called party number is set to the translated number. The original called number is the actual dialed number.

Figure 2 presents three call record signatures. To identify an outgoing call from A, it is necessary to observe the IAM sent from A's SSP and note its CIC value. Next, it is necessary to observe an answer message (ANM) sent to A's home SSP with the same CIC value. This IAM-ANM sequence reveals that a call is made from A to B (Signature 1).

A similar IAM-ANM message sequence identifies a call from B to A (Signature 2). An IAM is sent to A's SSP with the called party number set to A. An ANM is then returned from the SSP with the same CIC

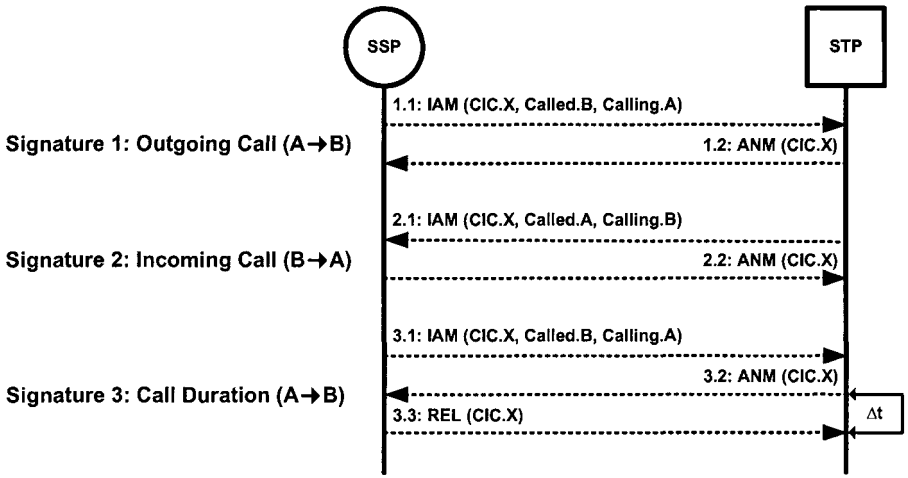


Figure 2. Basic call record signatures.

value. Since SS7 links are bidirectional, the direction of a message is inferred from its originating and destination point code parameters.

Call duration may be computed using IAMs and RELs or correlating IAMs, ANMs and RELs. The first method observes the release (REL) message and correlates it with the appropriate IAM. The call duration is then estimated based on the time difference between the generation of the IAM and the receipt of the REL. Because an IAM is generated when a number is dialed, this time difference overestimates the call duration (it includes the time spent waiting for the phone to be answered). To address this discrepancy, the second method also looks for ANMs with matching CICs. The time difference is calculated only after the called party answers and an ANM message is returned (Signature 3).

Table 2. Advanced call record data from call setup messages.

Call Record Data	ISUP Message
Unanswered Calls	IAM, ANM, REL
User Busy Failure	IAM, REL
User Release Direction	IAM, REL
Preemptive Release	IAM, ANM, REL

Analysis of signaling messages yields information about unanswered calls, calls terminated due to busy signals, preemptive hang ups by the caller, and preemptive hang ups by the receiver. Table 2 lists the ISUP messages that provide this data. Note that ISUP data are not collected by service providers and are, therefore, not available to law enforcement

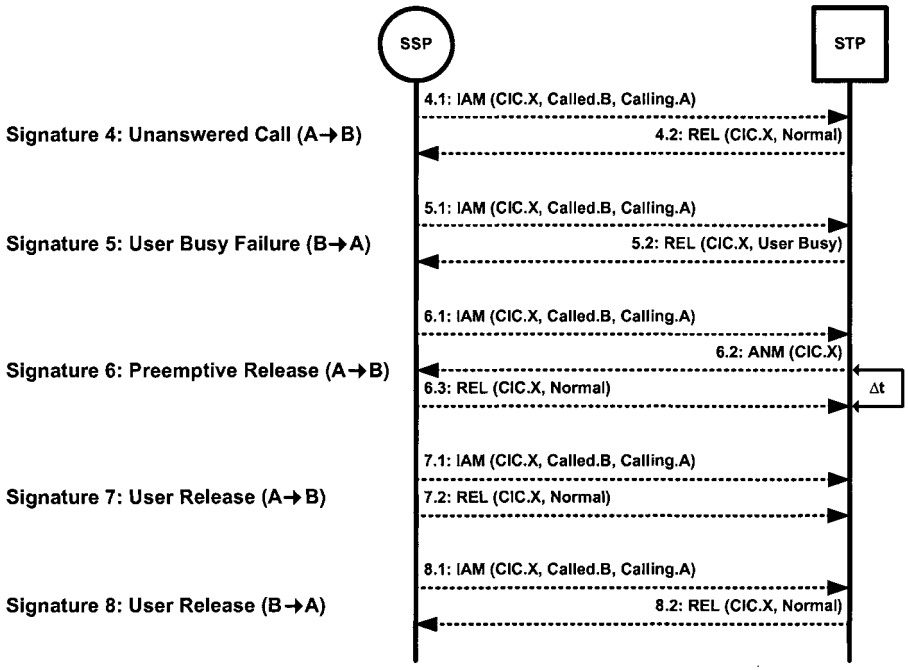


Figure 3. Advanced call record signatures.

agencies. For example, if an investigator has call records that contain only completed calls, and there is a record of a single call to an implicated phone number, the defendant could argue that he dialed the incorrect number. On the other hand, if the investigator has a record of all attempted calls, including numerous failed attempts to the implicated number before and after the completed call, it would be difficult to argue that a dialing error occurred.

Figure 3 presents five advanced call record signatures involving REL messages, whose cause code parameter indicates the reason for call termination. This parameter specifies whether a call completes normally, fails due to a busy signal, or ends because no one answers.

An unanswered call occurs when a caller hangs up before the phone is answered. Unanswered calls are detected by checking for IAMs, ANMs and RELs, even though the signature only involves an IAM and a REL (Signature 4). The presence of an ANM means that the call has been answered; therefore, an unanswered call does not have an ANM. The REL cause code is set to normal clearing because the caller hangs up and terminates the call normally.

Table 3. Storage requirements (full capture).

Links	Storage/Day
1	4.84 GB
100	483.84 GB
1,000	4.84 TB
10,000	48.38 TB

When a call is placed and a REL is received in response, the call does not complete. The attempted call is detected by correlating an IAM and REL message. The signature for a busy failure is an IAM followed by a REL with cause code set to user busy (Signature 5).

A preemptive release occurs when the caller hangs up immediately after the receiver answers – common behavior for pranksters and stalkers. The signature (Signature 6) is an IAM-ANM pair, followed by a REL within a short time period (1 to 3 seconds). Preemptive releases are also associated with call fraud. Many network surveillance devices record long distance calls with durations of a few seconds or less [9]. However, preemptive releases are distinct because hang ups occur in the initiating direction.

To determine who hangs up first, it is necessary to check the originating point code (OPC) and destination point code (DPC) of the REL message on the originating switch's SS7 link. If the OPC matches the originating switch's point code, then the caller hung up first (Signature 7). On the other hand, if the DPC of the REL matches the originating switch's point code, then the receiver hung up first (Signature 8). In both cases, the REL cause code is normal clearing.

4. Data Storage Requirements

As described in Section 2.3, the three options for collecting signaling traffic are: (i) full capture, (ii) protocol-specific capture, and (iii) message-specific capture. This section quantifies the storage requirements for each technique and presents a post-capture data processing technique that minimizes long-term storage costs.

Each SS7 link has a 56Kbps capacity. The storage requirement per link is also 56 Kbps (because fill-in messages are sent when regular messages are not transmitted). Table 3 lists the storage requirements for various numbers of links. The full capture technique is not feasible because most phone companies operate between 1,000 and 10,000 links.

Capturing only ISUP messages is a more reasonable approach because ISUP messages are mainly used for call setup. However, estimating the

storage required is difficult for three reasons. First, most – but not all – ISUP traffic is used for call setup. Second, all calls do not use the same number of messages. The number of messages depends upon the number of intermediate switches (nodes) along the voice path; typically, no more than six nodes are involved. Third, ISUP messages contain many optional parameters, most of which are never used. Therefore, the storage computation assumes that all ISUP messages are used for call setup and that these messages have only the optional parameters that are most commonly used.

Table 4. Storage requirements (ISUP message capture).

Number of Calls	2 Nodes	3 Nodes	4 Nodes	5 Nodes	6 Nodes
1 Call	256 B	512 B	768 B	1.024 KB	1.28 KB
1 Million Calls	256 MB	512 MB	768 MB	1.024 GB	1.28 GB
100 Million Calls	25.6 GB	51.2 GB	76.8 MB	102.4 GB	128 GB
1 Billion Calls	256 GB	512 GB	768 GB	1.024 TB	1.28 TB

In the simplest case, only two nodes are required to set up a call. Ten ISUP call setup messages are involved: an IAM (43 Bytes), ACM (17 Bytes), ANM (15 Bytes), REL (19 Bytes) and RLC (14 Bytes); each message is sent on two links. Table 4 indicates the storage requirements for various numbers of calls depending on the number of nodes involved in call setup. These figures incorporate 4-byte message timestamps. But they also assume that (aside from call setup) no other ISUP messages are sent. Therefore, the actual storage requirements could be 5% to 10% higher than specified in Table 4.

Table 5. Storage requirements (IAM-REL-ANM message capture).

Number of Calls	2 Nodes	3 Nodes	4 Nodes	5 Nodes	6 Nodes
1 Call	178 B	356 B	534 B	712 B	890 B
1 Million Calls	178 MB	356 MB	534 MB	712 MB	890 MB
100 Million Calls	17.8 GB	35.6 GB	53.4 MB	71.2 GB	89 GB
1 Billion Calls	178 GB	356 GB	534 GB	712 GB	890 GB

Since some ISUP messages, e.g., REL and RLC, contain redundant data, the storage requirements can be reduced by only capturing messages with useful data. Table 1 indicates that valuable call record data is contained in IAM, ANM and REL messages. The storage requirements for capturing these three types of messages are presented in Table 5.

Note that this technique has some inefficiencies. Often, a message traverses several links before it reaches its destination. As a result, identical messages are stored multiple times. It might seem reasonable to modify surveillance devices to store only single copies of a message. Unfortunately, the computational overhead outweighs any storage benefits.

However, post-capture data processing can significantly reduce the storage requirements without adversely impacting computational costs. The first step is to use the full ISUP capture or multiple-message capture technique. Next, the messages are analyzed and only the relevant fields from message sequences are retained.

Table 6. Post-capture call record attributes.

Attribute	Message	Size
OPC	IAM	3 B
DPC	IAM	3 B
Called Number	IAM	5 B
Calling Number	IAM	5 B
Call Start Time	IAM	2 B
Call Duration	IAM, ANM, REL	2 B
Received Answer Flag	IAM, ANM	1 B
Busy Failure Flag	IAM, REL	1 B
Preemptive Release Flag	IAM, ANM, REL	1 B
User Release Direction Flag	IAM, REL	1 B

Table 6 lists the important attributes contained in ISUP post-capture call records along with their size. Originating and destination point codes (OPCs and DPCs) are collected to assist in searching records. To track all calls to a particular number, one could naively search through all the records for the called and calling party numbers. However, the search effort can be significantly reduced by only looking for OPCs or DPCs with the point code of the home switch of the target number.

Table 7. Storage requirements for 1 billion calls.

Data Collection Technique	Storage
Full ISUP Capture	768 GB
IAM-REL-ANM Message Capture	534 GB
Post-Capture Data Processing	26 GB

Table 7 compares the storage requirements for the three techniques based on one billion phone calls. The average of each of the node costs is used for computing the storage requirements for the full ISUP capture

and the IAM-REL-ANM capture techniques. Note that the IAM-REL-ANM message capture technique requires significantly more storage than the post-capture data processing technique. This is because ISUP messages contain only a few parameters that are required for creating useful call records, and the same messages are repeated several times along the call setup path. The post-capture data processing technique is promising because it eliminates extraneous and redundant data.

5. Conclusions

Call records generated from signaling messages can be very valuable in criminal investigations. Signaling messages provide data about phone calls – not the content of phone conversations. Therefore, collecting and analyzing signaling messages may not be subject to the same legal restrictions as recording voice conversations. Signaling messages are routinely examined by service providers to maintain quality of service, debug network problems and generate billing records. Service providers could use these same messages to obtain data about user calling patterns that could be provided to law enforcement when authorized. This data can be collected using existing surveillance equipment [8, 9], and it can be analyzed and stored at little additional cost.

Collecting and archiving massive quantities of call setup traffic raises security and privacy concerns. Obviously, legal and technical safeguards must be implemented to prevent abuses by service providers and law enforcement personnel. Still, storing all call records for prolonged periods of time is almost Orwellian.

Hashing techniques can be employed to allay these concerns. Instead of storing call records, their hash values are computed and saved. Then a law enforcement agent could query the hash values, for example, to check if a certain phone number was called from a target phone number. Bloom filters [5] permit (precise) negative responses to queries, but affirmative responses only have an associated probability. These techniques provide numerical confidence levels for affirmative responses. Such information could assist investigations by ruling out certain theories or corroborating other evidence. The hashing techniques would not provide the same detail and level of confidence as storing (and querying) all relevant data – but they would certainly protect privacy.

A major technical challenge is introduced by the convergence of the public telephone infrastructure with VoIP networks. This convergence introduces new protocols and signaling points to the infrastructure [6]. Because these networks still interface using the SS7 protocol, monitoring the signaling links that connect VoIP systems to public telephone

networks can cover a significant amount of call traffic. However, VoIP calls routed exclusively over the Internet are difficult – if not impossible – to monitor because of the decentralized nature of pure VoIP networks.

References

[1] American National Standards Institute (ANSI), *T1.113-1995: SS7 Integrated Services Digital Network (ISDN) User Part*, New York, 1995.

[2] American National Standards Institute (ANSI), *T1.111-1996: SS7 Message Transfer Part (MTP)*, New York, 1996.

[3] American National Standards Institute (ANSI), *T1.112-1996: SS7 Signaling Connection Control Part (SCCP)*, Institute, New York, 1996.

[4] American National Standards Institute (ANSI), *T1.114-1996: SS7 Transaction Capabilities Application Part (TCAP)*, New York, 1996.

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

[6] O. Ibe, *Converged Network Architectures*, John Wiley, New York, 2002.

[7] T. Russell, *Signaling System #7*, McGraw-Hill, New York, 2000.

[8] Tekelec, Integrated Application Solutions (www.tekelec.com).

[9] Tektronix, GeoProbe (www.tek.com).

[10] Telcordia, *GR-82: Signaling Transfer Point (STP) Generic Requirements*, Morristown, New Jersey, 2001.

[11] Telcordia, *GR-1241: Supplemental Service Control Point (SCP) Generic Requirements*, Morristown, New Jersey, 2001.

[12] R. Thompson, *Telephone Switching Systems*, Artech, Norwood, Massachusetts, 2000.

[13] J. Van Bosse, *Signaling in Telecommunication Networks*, John Wiley, New York, 1997.

IV

PORTABLE ELECTRONIC DEVICE FORENSICS

Chapter 16

FORENSIC ANALYSIS OF MOBILE PHONE INTERNAL MEMORY

Svein Willassen

Abstract Modern mobile phones store data in SIM cards, internal memory and external flash memory. With advanced functionality such as multimedia messaging becoming common, increasing amounts of information are now stored in internal memory. However, the forensic analysis of internal memory, including the recovery of deleted items, has been largely ignored. This paper presents two methods for imaging the internal memory of mobile phones. The methods are applied on several popular models to recover information, including deleted text messages.

Keywords: Digital evidence, mobile phones, internal memory

1. Introduction

Mobile phones have become the primary tool for personal communication and they frequently contain information that may have evidentiary value. It is, therefore, vital to develop forensically sound methods for extracting and analyzing digital evidence from mobile phones.

This paper presents two methods for imaging the internal memory of mobile phones. The methods are applied on several popular models to recover important evidence, including deleted text messages.

2. Mobile Phone Memory

Mobile phones are digital media; therefore, in principle, they have the same evidentiary possibilities as other digital media, e.g., hard drives. Deleted information can be extracted from a mobile phone in much the same way as it is obtained from a hard drive. Like other digital media, mobile phone memory is fragile and is easily deleted or overwritten.

Moreover, since a mobile phone is a complex, compact device, great care should be taken while attempting to extract evidence.

Considerable information is stored in mobile phones [19]. In addition to telecommunications-related information, modern phones contain images, sound files, multimedia messages, WAP/web browser history, email, calendar items and contact lists. SMS (short message service) text messages, which often contain useful information, are stored on the transmitting and receiving phones. Indeed, recovering deleted SMS messages was the main motivation of this research.

The following subsections describe the principal components of mobile phones that store information of evidentiary value.

2.1 Subscriber Identity Module

The advent of digital mobile telephony in the 1990s created a need for local storage in mobile phones. GSM, the most popular digital mobile phone system, mandates a SIM (Subscriber Identity Module) to be present inside each GSM device. A SIM card incorporates a processor and EEPROM memory. The SIM architecture is also used in at least one 3G system (USIM in UMTS).

SIM cards contain subscriber information and encryption keys for secure communications; they also store contact lists and text messages. Much of this information, including deleted text messages, can be recovered depending on the mobile phone model [19].

2.2 Internal/External Memory

Rigorous SIM specifications have prevented SIM cards from being used as general purpose memory storage devices. Therefore, as manufacturers implemented new functionality that required additional storage, mobile phones were equipped with internal memory, e.g., for storing missed and received calls, calendar events, text messages and contacts.

The first models used serial EEPROM chips as internal memory. However, the use of mobile phones as cameras and music players has led manufacturers to add external flash memory (e.g., SD, MMC, CF cards). External memory cards can be analyzed using commonly available tools [3, 4, 20].

3. Mobile Phone Architecture

Figure 1 presents the basic mobile phone architecture. The CPU performs all computational tasks, including controlling the communications circuits. The CPU uses the RAM for temporary storage during phone

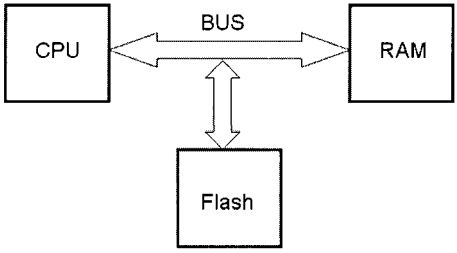


Figure 1. Mobile phone architecture.

operation. The RAM is a separate integrated circuit (IC) or it may be packaged with the CPU in a single IC.

Mobile phones also have secondary non-volatile storage for user and communications data that persist after they are powered down. Most commonly, secondary storage is implemented as separate flash memory integrated in the phone.

The CPU also communicates with the SIM card and additional external storage media, if present. Often, a special unit is present to control power usage, especially by the wireless transceiver.

4. Internal Memory Analysis

Standard methods do not exist for analyzing internal memory. Currently, information is extracted via AT commands using cable, infrared or bluetooth connections to a phone. GSM specifies a standard command set for extracting certain data [1]. Model- and manufacturer-specific data may be extracted using proprietary commands. Several software packages are available for this purpose, e.g., Oxygen Phone Manager, Paraben Cell Seizure and TULP2G [13, 14, 16].

AT commands use the phone's operating system to extract information. A major limitation is that deleted information may not be obtainable using these commands.

In some cases, potentially important evidence is deleted when a phone detects the presence of a new SIM card on start-up. Also, a phone might require a PIN code for it to be used with a specific SIM card. Therefore, if a phone is on when it is seized, it is suggested that it be kept on until the analysis is complete.

A jamming device or Faraday cage is recommended to ensure that a phone will not communicate with the network, and possibly have its memory modified. However, as discussed later, this may not prevent memory contamination. Moreover, if a phone is prevented from receiv-

ing signals, it will continuously probe the network for connectivity, consuming more energy and ultimately exhausting its battery.

Another method is to analyze a phone using a "cloned" SIM card. But this still involves turning the phone on and reading its memory using the operating system.

As long as a phone is on, it is possible for deleted information in its memory to be overwritten. Therefore, it is necessary to develop forensic methods for analyzing "dead" mobile phones. These methods would enable investigators to turn off mobile phones upon being seized, while ensuring that the internal states of the devices are maintained.

The following sections describe two methods for analyzing "dead" mobile phones: forensic desoldering that removes and images memory ICs, and imaging memory using a built-in test methodology. A third method is also possible – loading software into system RAM to read the flash memory through the system interface. But this method requires software specific to each mobile phone, and is not discussed in this paper.

5. Forensic Desoldering Method

The first method proposed for imaging the internal memory of mobile phones is to desolder the memory circuits and read the data off the chip.

5.1 Ball Grid Array Technology

Ball Grid Array Technology (BGA) is used to mount SMT (Surface Mount Technology) components on printed circuit boards (PCBs). Each chip has an array of pads on its lower side on which small balls of solder are placed. A chip is mounted by placing it on matching pads on the PCB and using reflow soldering [11] to bond the melted balls to the PCB pads. BGA technology uses less space on the PCB. Conventional ICs are limited not by chip area, but by the area for pins. BGA enables the entire area on the bottom of a chip to be used for signaling.

BGA presents difficulties to the forensic investigator. With conventional surface mounting, probes can be attached to the pins to read a chip. Thus, an investigator may access the memory contents without destroying the unit. However, BGA bonds the chip to the PCB. Individual probes cannot be attached to the chip, and the chip must be desoldered from the PCB before it can be read.

Desoldering must be performed with great care to avoid damaging the memory circuits. Melted solder residue must be removed so it does not short circuit the chip pads. Then, the chip is restored to its original state using a "reballing process." Alternatively, its contents are read by connecting directly to the pads.

5.2 Reballing

Many device readers require a chip package to be intact with its solder balls in place. Reballing restores the solder balls on the pads on the package's lower side. A special reballing stencil is used; this stencil has small holes that match the size and placement of the pads. The stencil is aligned so that each hole matches a pad. Solder paste is then applied on the stencil and evenly distributed until all the holes are properly filled. Finally, reflow is applied, which causes the solder paste to form balls that are fixed to the chip pads.

BGA reballing requires that the stencil match the ball size, inter-ball distance (pitch) and ball configuration of the chip. Since mobile phones have packages with different sizes and configurations, obtaining the right stencil can be quite difficult.

5.3 Reading Memory

A device programmer may be used to read memory circuits. A variety of adapters are available for device programmers, e.g., adapters for DIP or SOIC packages with up to 40 leads, or adapters for Intel 28F640 chips. In addition, the correct software is needed to read a chip as the pin configurations can vary. Fortunately, most manufacturers supply software with their device programmers that enables a variety of devices to be read. Many devices have built-in checksum capabilities to detect inconsistencies during the reading process; this capability must be supported by the software.

A device programmer may be used to read BGA circuits when conducting a forensic analysis of mobile phone memory. The adapter pins must match the ball size, pitch and layout. Adapters use either Y-shaped springs or spring-loaded pogo pins to establish chip connections. A Y-shaped pin must have a ball in place to establish a connection. A pogo pin can be used directly on the chip pads without balls. From the forensic point of view, spring-loaded pogo pins are better because reballing is avoided.

5.4 Desoldering

Equipment used for desoldering ranges from simple soldering irons to massive reflow ovens. The most important consideration from the forensic point of view is potential damage to the unit. An IC cannot endure more than a certain number of heatings to reflow temperature. Intel BGA circuits are guaranteed to endure reflow three times [7, 8]. Since a PCB with SMT components on both sides is assembled using two

reflows, only one heating to reflow temperature is available in a forensic investigation. This is not enough if both desoldering and reballing must be performed. However, experience indicates that components will survive more than the specified maximum number of reflows.

Proper temperature profiles must be maintained so as not to damage the unit. Since conventional mounting technology is not used, the entire chip must be heated. The maximum temperature should be nominally above the solder reflow temperature, and the gradient should not be too steep. Specifically, the temperature should have a "tent" profile with a peak of 210°C, which is reached in 3-4 minutes [10]. Pre-baking is recommended to remove moisture in the unit. Otherwise, heating may cause a "popcorn-effect," whereby water vapor causes the circuit substrate to crack. Moisture sensitivity and conditioning requirements are provided in [17].

These requirements rule out the use of hand soldering tools. Forensic desoldering should only be performed using an automatic soldering station with temperature gradient programming and automatic component removal capabilities. Such a soldering station typically employs a combination of convection heating using hot air and infrared (IR) radiation heating. It has a temperature-controlled hot air nozzle and can be programmed to follow specific heating curves.

5.5 Experimental Results

Several different phone models were dismantled, desoldered and imaged. The results are summarized in this subsection.

The mobile phone units were dismantled before desoldering. Manufacturers often provide dismantling instructions for models upon request. All dismantling should be done in an electrostatically safe environment as the exposed components are sensitive to electrical discharges.

One or more PCBs are exposed upon dismantling a unit. Due to space constraints, most phones have a single circuit board with surface mounted components on both sides. A circuit board contains several inner layers that are not exposed; therefore, it is not possible to trace the leads by examining the board.

The boards that were examined had 5-10 surface mounted BGA ICs. The ICs were identified by referring to databooks and/or manufacturer websites.

The flash memory circuits were desoldered by pre-baking them for 24 hours at 80°C to remove moisture. Next, the circuits were desoldered using a hot air soldering station with IR preheating. The temperature profile had a peak of 220°C, which was reached in 5 minutes. The

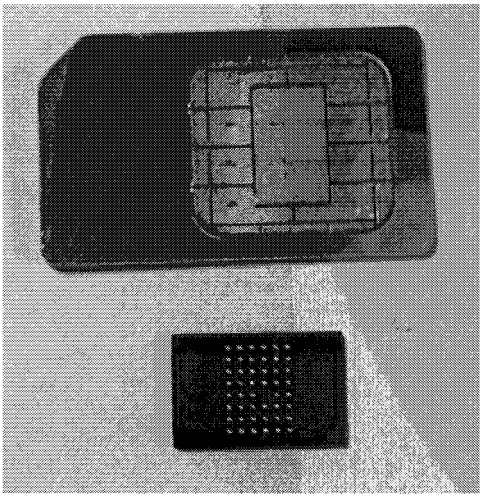


Figure 2. Reballed BGA package with 0.75mm pitch.

chips were removed from the PCB at the maximum temperature using a robotic arm with a vacuum sucker mounted on the soldering station.

Using the vacuum sucker, the chip may be removed with the solder balls intact at a temperature where the solder is marginally above the melting point of 183°C. However, this practice is risky as it is difficult to measure the exact temperature of the solder. Removing a circuit at too low a temperature can rip the pads off, permanently damaging it. Consequently, it was decided to use a temperature of 220°C, which was sufficiently above the melting point (210°C).

Removing the circuits from the board exposed the chip pads on their undersides; these were partially covered with solder residue. The residue was removed very efficiently using a temperature-controlled hot air soldering station and a soldering wick. Note that removing solder residue requires the unit to be heated, which should be done with great care. Finally, reballing was performed for the packages for which stencils were available. Figure 2 shows a reballed BGA package; a SIM card is shown above it for size comparison.

The chips from the mobile phones were mounted on a device programmer, and the contents read using the supplied software. An adapter was not available for some of the packages. For others, the built-in checksums indicated errors during the reading process; it could not be determined if the errors were due to damage caused by desoldering-reballing or for other reasons. A complete memory dump was obtained in most cases, and these files were be analyzed forensically.

6. Embedded Test Technology Method

Most electronic devices are built using embedded test technology. The reason is that manufacturers need automated systems for testing device elements and interconnections, otherwise the quality of the devices cannot be guaranteed.

The current trend in test technology is to use "boundary-scanning." This technique uses a shift register on an integrated component to probe and set the pins of the component. The shift register can be loaded serially from a test access port (TAP) accessible as test points on the device. If the device CPU supports boundary-scanning, the bus can be controlled from the TAP. It is then possible to read or program the memory attached to the bus. This technique is called "in-system programming." The JTAG standard (IEEE 1149.1) [5] governing boundary-scan implementations facilitates the interoperability of components from different manufacturers [6].

6.1 JTAG In-System Programming

JTAG supports in-system programming and reading. If the memory device supports JTAG itself, the TAP of the memory device can be addressed through the JTAG interface. For each memory address, the in-system programmer sets the address value on the device's address bus by shifting in the correct bits through the boundary-scan register. The data bus will then contain the content of the desired memory address and can be read by shifting the value of the boundary-scan register back through the JTAG chain. The entire memory contents can be read by probing memory addresses in this manner.

If the memory device does not support JTAG, in-system programming can be performed using the TAP of another device connected to the memory system bus. This device could be the system processor, which usually has a direct connection to the memory via the system bus. The memory is then read by manipulating the boundary-scan register of the CPU, setting an address on the bus, and reading data through the CPU TAP.

6.2 JTAG Memory Reading Challenges

Note that JTAG is not a bus standard but a standard for component TAPs that facilitates interconnections between components on a board. However, IC manufacturers determine the pin configuration and functionality, and the boundary-scan register configuration. Also, PCB designers can decide if JTAG ports should be interconnected or be acces-

sible from test points on the board. The designer can decide not to use JTAG, leaving the test pins unconnected. If a design uses BGA circuits, attaching a probe to the test pins is impossible and reading memory through JTAG is not an option. However, since manufacturers would otherwise have no way to test their designs, it is common for them to implement JTAG in conjunction with BGA circuits.

Before attempting to use JTAG to read memory, it is necessary to identify the system processor and memory circuits, and their bus connections. Otherwise it would be impossible to find the correct bits in the boundary-scan register. The JTAG test points on the printed circuit board must be located and their signals determined. Also, the memory read/write protocol must be known. Moreover, the correct voltage must be determined; using too high a voltage can damage the circuits.

The voltage can be determined by measurements on a live board. The memory protocol is generally available at the manufacturer's website. However, identifying the bus connections and determining the JTAG test point signals are very difficult without complete documentation. Such documentation is rarely available for mobile phones. Also, implementations for reading memory via JTAG differ from phone to phone. Even a small configuration change to a model may require a separate JTAG interface.

6.3 Experimental Results

Experiments were conducted on an Nokia 5110 mobile phone. The Nokia 5110 was chosen because it was inexpensive, and its service manual and schematics were available. The 5110 is an older model, and its internal memory only stores a few items. Nevertheless, the results can be extended to new mobile phone models.

The 5110 manual was examined for indications of JTAG implementation [12]. The CPU, listed as `MAD2`, had pinouts for `JTRst`, `JTClk`, `JTDI`, `JTMS` and `JTDO`. These pins were coupled in a line `JTAGEMU` onto a connector, listed in the schematics as "not assembled." A 5110 was disassembled, and the test points corresponding to this connector on the system board were found. The connections between the test points and the CPU were partly visible, enabling the test points to be identified.

Voltage measurements indicated that the test points were connected to the CPU's JTAG interface. Test wires were carefully soldered to the connectors using very thin wires from an 80-pin EIDE cable (see Figure 3). Since the test points were very small, the soldering was difficult, but sufficiently thin wires and soldering pate resulted in success. The risk of

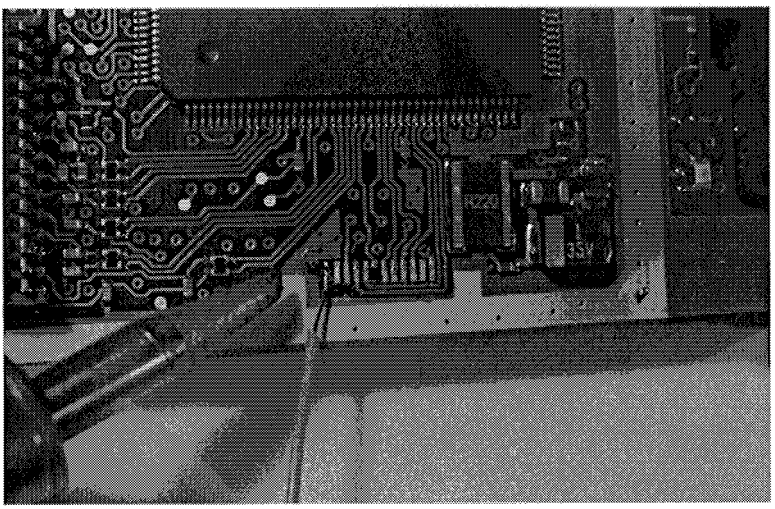


Figure 3. Thin wire attached to a JTAG test point on a Nokia 5110 board.

damaging circuits is lower than for BGA chip desoldering. However, it was tedious to obtain the proper connections without short circuits.

The 5110 was then connected to a computer through the JTAG interface using a breadboard. The Chameleon POD programmable JTAG interface was used in the experiment. When connecting the phone to the interface, the voltage level of the phone system board and the interface should be considered. The 5110 system board uses 2.7V technology, which is in the acceptable range for the Chameleon POD.

Next, the JTAG interface was connected to a Linux machine with the open-source JTAG Tools package [15]. This package allows for device connections via various adapters, including several that can be programmed on the Chameleon.

After some experimentation, a connection with the 5110 processor TAP was established. Although the service manual indicated that the processor design was based on ARM7 and documentation on its JTAG interface was available, the experiment treated the TAP as a "black box," as this would be the case in most situations. JTAG Tools allows for black box analysis via the `discovery` command. This function cycles a `1` through the JTAG chain and detects the number of available TAPs and the length of their instruction register (IR). For the 5110, the IR was determined to be 12 bits long with only one JTAG TAP connected (the processor). The software continued the analysis by probing all possible values of the IR by cycling a `1` through the JTAG chain and detecting the data register length for each instruction. Thus, the values for different instructions were determined. In all, 4096 different possibilities were

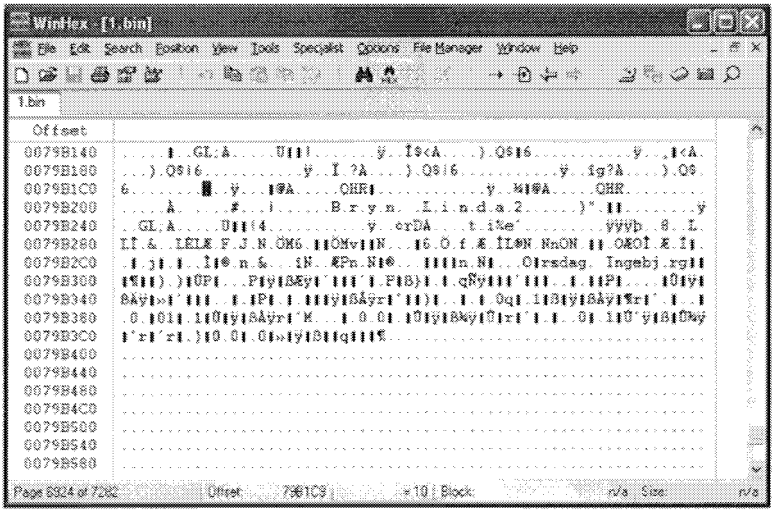


Figure 4. Sony Ericsson T68i memory dump with contact and calendar entries.

tested. Eventually this process resulted in the discovery of commands that could be used to set and probe the boundary-scan register. The command probing process does not change the boundary-scan register and can, therefore, be executed without risk of altering memory content.

Details for setting the boundary-scan register bits to probe the Sharp L28F800BE-TL85 flash memory were obtained from the schematics. These details were implemented in definition files for JTAG Tools, and the memory was then read using the `readmem` command.

7. Memory Analysis

After applying the forensic desoldering and JTAG techniques, the contents of flash memory were available as a binary file. This file was analyzed using a standard hex editor (WinHex [20]).

The mobile phone's Intel 28F640W18T chip yielded an 8 MB memory dump. At first glance, the dump appeared to contain useless information. However, further analysis revealed useful information, such as operating system error messages. This confirmed that the chip was read successfully and that the contents could be analyzed. Further examination showed that the first 4-5 MB of the dump contained binary data of high entropy, corresponding to system software. The remaining dump contained large areas filled with hex `FF` values and smaller areas filled with user data.

Analysis of the data areas revealed GIF images, JPEG images, phone numbers, calendar items and text messages in TPDU format [2]. The

images were extracted and viewed on a computer. The text included contact and calendar items (see Figure 4). Other information associated with these items, e.g., phone numbers and timestamps, had to be obtained by interpreting the binary data.

Memory dumps from other mobile phones were analyzed with similar results. However, much work remains to be done to identify evidentiary items in memory dumps and present them clearly.

7.1 Experimental Results

An important goal was to discover if deleted items could be recovered by analyzing memory dumps and, in particular, if deleted text messages could be recovered. Several experiments were performed that involved reading the internal memory after entering and deleting data via the phone's operating system. The limitations of current memory reading methods made it difficult to conduct more than a few tests. Another difficulty was the lack of a software tools for identification and interpretation of TPDU messages in memory dumps.

Nevertheless, the experiments indicated that text messages were still present in memory after they had been "deleted" using the operating system. Images, MMS messages, calendar items and contacts were also found after they had been deleted. In addition, data pertaining to SIM cards used with the mobile devices were found in some cases.

Interestingly, the search for deleted text messages revealed that memory managers were used to dynamically reallocate memory during phone use. The presence of memory managers has significant ramifications to the forensic analysis of mobile phones.

7.2 Implications

Currently, the forensic analysis of a mobile phone involves keeping the phone on after it has been seized. The memory contents are then analyzed by connecting the phone to a computer using a cable and reading data via the phone's operating system. Since the phone must be on during this activity, it is recommended that the phone be kept on after it is seized.

However, if deleted data can be recovered and if memory management software exists on a mobile phone, keeping the phone turned on for indefinite periods after seizure may not be appropriate. The memory manager might automatically reorganize the memory, potentially overwriting evidence at any time, especially the deleted items.

Techniques used for mobile phone analysis may ultimately mirror the evolution of forensic procedures for computers. Originally, computer

forensic analysis was conducted by booting the operating system and searching for evidence. Now it is recognized that this can destroy evidence; therefore, hard drives are imaged before being analyzed. Similar progress will likely occur in the case of mobile phones. Since mobile phones have much less memory than hard drives, there is a higher risk of losing evidence due overwriting if the phone is kept on.

8. Conclusions

Forensic desoldering and JTAG are promising methods for imaging the internal memory of mobile phones. The methods are technically demanding, and require specialized equipment and skills. However, the resulting memory dumps, which can be analyzed with standard hex editors, contain information of evidentiary value.

The presence of memory managers in modern cell phones raises an important issue. If a phone is kept on after being seized, there is a chance that the memory manager might move memory blocks and/or overwrite important evidence. This suggests that mobile phones must be turned off right after they are seized. Their SIM cards and external flash memories should be removed and analyzed separately. Then, their internal memories should be imaged and analyzed using the methods proposed in this paper.

The results of this work are applicable to other personal electronic devices (e.g., PDAs and GPSs) as well to vehicular navigation systems and other embedded devices [18]. In particular, using physical memory extraction instead of operating system commands facilitates the recovery of deleted information.

References

[1] 3G Partnership Project, *AT Command Set for GSM Mobile Equipment (ETSI ETS 300.642)*, October 1998.

[2] 3G Partnership Project, *Technical Realization of Short Message Service (ETSI ETS 123.040)*, September 2003.

[3] B. Carrier, Sleuthkit (www.sleuthkit.org).

[4] Guidance Software, EnCase (www.encase.com).

[5] IEEE, *IEEE Standard Test Access Port and Boundary-Scan Architecture (IEEE 1149.1)*, Piscataway, New Jersey, 2001.

[6] Intel Corporation, *Designing for On-Board Programming Using the IEEE 1149.1 (JTAG) Access Port*, Santa Clara, California, 1996.

[7] Intel Corporation, *Ball Grid Array Packaging Handbook*, Santa Clara, California, 2000.

[8] Intel Corporation, *Intel Wireless Communications and Computing Package User's Guide (Version 1.2)*, Santa Clara, California, May 2004.

[9] G. Le Bodic, *Mobile Messaging Technologies and Services: SMS, EMS and MMS*, John Wiley, New York, 2005.

[10] N. Lee, *Reflow Soldering Processes and Troubleshooting: SMT, BGA, CSP and Flip Chip Technologies*, Elsevier Science, Oxford, United Kingdom, 2001.

[11] H. Manko, *Solders and Soldering*, McGraw-Hill, New York, 2001.

[12] Nokia Corporation, *Nokia NSE-1 Series Cellular Phone Service Manual*, Salo, Finland, March 1998.

[13] Oxygen Software, Oxygen Phone Manager (www.oxygensoftware .com).

[14] Paraben Corporation, Cell Seizure (www.paraben.com).

[15] SourceForge.net, JTAG Tools (openwince.sourceforge.net/jtag).

[16] SourceForge.net, TULP2G: Forensic framework for extracting and decoding data (tulp2g.sourceforge.net).

[17] B. Vaccaro, R. Shook and D. Gerlach, The impact of lead-free reflow temperatures on the moisture sensitivity performance of plastic surface mount packages, *Proceedings of the International Conference of the Surface Mount Technology Association*, 2000.

[18] R. van der Knijff, Embedded systems analysis, in *Handbook of Computer Crime Investigation: Forensic Tools and Technology*, E. Casey (Ed.), Elsevier, London, United Kingdom, pp. 315-360, 2004.

[19] S. Willassen, Forensics and the GSM mobile telephone system, *International Journal of Digital Evidence*, vol. 2(1), 2003.

[20] X-Ways Software Technology, WinHex: Computer Forensics and Data Recovery Software (www.x-ways.net/winhex).

Chapter 17

IMAGING AND ANALYSIS OF GSM SIM CARDS

Christopher Swenson, Gavin Manes and Sujeet Shenoi

Abstract Cellular phones are becoming ubiquitous. As of March 2005, there were more than 180 million cellular subscribers in the United States, over 60% of the population. Cellular devices invariably contain information that can aid criminal investigations. Nevertheless, extracting evidence from cellular phones is quite uncommon in the United States. The principal reasons are the lack of awareness and training on the part of law enforcement agents and the limited availability of inexpensive tools for extracting and analyzing evidence. This paper describes a toolkit for extracting and analyzing data from SIM cards, which are used for cryptographic authentication, key generation and data storage in GSM cellular phones.

Keywords: Digital forensics, cellular phones, GSM, SIM cards

1. Introduction

Tools for recovering evidence from computer hard drives and electronic devices are not new in the field of forensics [1, 7, 11]. Nevertheless, extracting evidence from cellular phones is quite uncommon in the United States. The principal reasons are the lack of awareness and training on the part of law enforcement agents and the limited availability of inexpensive tools for extracting and analyzing evidence.

This paper describes the design and implementation of a toolkit for extracting and analyzing data from SIM cards, which are used for cryptographic authentication, key generation and data storage in GSM cellular devices. The toolkit, which has been created specifically for law enforcement agents, will assist in bringing digital evidence from cellular devices to bear in criminal cases.

2. GSM Communications

The Global System for Mobile Communications (GSM) is a second-generation (2G) cellular phone protocol. GSM cellular phones have been selected as the focus of this research for two main reasons.

- GSM is completely standardized. GSM cellular devices are typically usable in any GSM-compliant country with few problems. Furthermore, GSM standards are available free-of-charge, which facilitates the development of forensic tools.
- Each GSM phone contains a smart card for cryptographic functions and data storage. Although GSM handsets differ considerably, the smart cards are standardized, allowing evidence to be extracted using common methods. Data stored on these smart cards can be vital to criminal investigations.

Until recently, GSM penetration in the United States was relatively limited. Most companies, e.g., Verizon, use different cellular networks, such as IS-136 [15] and cdma2000 [9]. However, Cingular and T-Mobile now provide mostly GSM service, and GSM is the fastest growing cellular network in the United States.

GSM cellular telephones incorporate a handset and a smart card. The handset, which is responsible for communicating with the Public Land Mobile Network (PLMN), stores user information and interfaces with the user. The smart card, called the Subscriber Identity Module (SIM), is housed inside the handset. The SIM stores user information and is also responsible for cryptographic authentication and key generation.

Although a considerable amount of information is stored on the handset, extracting it in a forensically sound manner is extremely difficult due to the variety of proprietary interfaces and the lack of documentation for each make and model. As such, the focus of this paper is on extracting and analyzing data from the SIM card rather than the handset.

3. Subscriber Identity Module (SIM) Cards

SIM cards are smart cards that support GSM communications. Smart cards were originally developed to address security issues with magnetic stripe cards that could be read and modified qute easily. The smart card solution incorporated a microprocessor to control access to the memory on the card, and to provide computational power for other operations, e.g., cryptography. The first smart cards deployed in the mid-1980s for use in telephone systems [14] met with immediate success. Smart card usage has increased as they have become more powerful and less expensive, which has enabled them to be used in modern cellular phones.

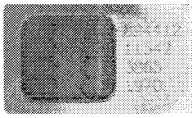


Figure 1. SIM Card (actual size: 25mm × 15mm).

Smart cards typically have two components: memory, which is usually in the form of an EEPROM chip, and a microprocessor, often used to access and protect the memory and to provide cryptographic functionality. These cards were originally made of plastic and were about the size of a credit card, but they have been getting smaller. Most smart cards communicate through a serial data channel using several contact pins. Recently, there has been a surge of contactless cards that communicate wirelessly with host devices.

SIMs are smart cards programmed to support specific GSM operations, notably cryptographic authentication and key generation. They also store some user information. Figure 1 shows a SIM card manufactured for Orange, a GSM provider in the United Kingdom.

4. SIM Card Forensics

During an investigation, it is necessary to ensure that all user information from a SIM card is retrieved. Also, non-user information, e.g., data related to the use of the SIM card as a cryptographic device, which may not be of much importance to the investigation must be retrieved (so long as it does not affect the integrity of the SIM card and user data). Finally, the integrity of the data must be maintained and should be verifiable. This is usually accomplished by computing and storing a series of hashes and checksums on the data.

4.1 SIM Card File Structure

SIM cards incorporate simple hierarchical file structures with certain classes of files used to organize and secure larger groups of files, providing directory-like functionality. Each file has a descriptor byte indicating the file's type, and a location byte that distinguishes individual files. Files can be elementary files, dedicated files or master files. Table 1 lists the different file types and the associated header numbers [6].

The Master File (MF) is a unique file present on all SIM cards. The MF acts as the root directory, and usually has a small number of elementary (data) files, with most files on the SIM card contained in directory-like objects called dedicated files (DFs). An Elementary File (EF) is a container for data, either in records or as a simple byte stream. Records

Table 1. SIM card file types.

Descriptor Byte (Hexadecimal)	File Type
3F	Master File (MF)
2F, 4F, 6F	Elementary File (EF)
5F, 7F	Dedicated File (DF)

can only be accessed in one of two modes for a given EF: "linear-fixed" mode, i.e., each record is accessed as a sequential list (appropriate for contact lists), and "circular" mode, i.e., access is based on recency using a queue (appropriate for call lists).

Table 2. Important SIM card files.

File Name/Location	Description
3F00 7F10 6F3A	Abbreviated Dialing Numbers
3F00 7F10 6F3C	Short Message Service storage
3F00 7F10 6F40	Mobile Subscriber ISDN
3F00 7F20 6F21	International Mobile Subscriber Identity

4.2 SIM Card Files

Table 2 lists the files found on a SIM card that may contain information valuable to investigations [6].

4.2.1 International Mobile Subscriber Identity. The International Mobile Subscriber Identity (IMSI) is a unique 15-digit decimal number, usually encoded as packed Binary Coded Decimal (BCD), that identifies each GSM network user. The IMSI is broken down into several digit groups. The first three digits correspond to the Mobile Country Code (MCC) [8], the next two or three digits constitute the Mobile Network Code (MNC), and the last nine or ten digits comprise the Mobile Subscriber Identification Number (MSIN). The MCC identifies the country where the IMSI is intended to be used, while the MNC identifies the service provider, e.g., T-Mobile, Cingular. Tables 3 and 4 list common MCCs and MNCs, respectively. The MSIN is used by the service provider to identify individual subscribers.

4.2.2 Mobile Subscriber ISDN. The Mobile Subscriber Integrated Services Digital Network (MSISDN) is the standard telephone

Table 3. Common mobile country codes.

Mobile Country Code	Country
208	France
234–235	United Kingdom
310–316	United States
330	Puerto Rico
332	Virgin Islands, U.S.
334	Mexico

Table 4. Common mobile network codes.

MCC	Mobile Network Code	Provider
310	090	Edge Wireless
310	15	BellSouth Mobility DCS
310	150	Cingular Wireless
310	20–27	T-Mobile
310	410	Cingular Wireless

number used to place or receive calls [4]. A SIM card can use several MSISDNs simultaneously – the elementary file has a linear-fixed structure, allowing a small number of MSISDNs to be allocated. However, in practice, it is common for a user to use one MSISDN at a given time.

4.2.3 Contact List. Contact list information is stored primarily in the Abbreviated Dialing Numbers (ADN) file. The ADN contains telephone numbers along with a small amount of text associated with each number, usually a name. Some cellular phones use special grouping information in the text area to assign different ring tones and to differentiate between home, mobile and work numbers.

4.2.4 Short Message Service. Short Message Service (SMS), a service offered by most wireless providers, allows subscribers to exchange short messages of a few hundred characters. This service is popularly called "text messaging." The message format, standardized in [2], contains a source messaging center (essentially a router), the source telephone number, and the message. The alphabet for encoding SMS messages typically has an uncompressed 7-bit format [3]. A message larger than the maximum number of characters is usually split into two or more messages. Normally, only incoming SMS messages are stored on SIM cards. Outgoing messages are stored only in the handset memory, if at all.

5. SIM Card Imaging

The standard forensics practice is to make an exact duplicate of the digital evidence and then perform the analysis on the copy. For SIM cards this is important, as SIM card storage is typically in the form of EEPROM, which has a limited number of read-write cycles. If an investigator were to perform all the analysis on the physical device, it is possible that the device could be permanently damaged. This would compromise the integrity of the evidence as well as undermine the ability to complete the investigation.

This section describes the design of an imaging tool for SIM cards. The tool uses standard smart card readers to obtain data from SIM cards according to the guidelines identified in the previous section.

Tools are available for acquiring data from SIM cards. However, they do not meet the stringent forensic requirements for data acquisition. For example, the `sim_scan` tool gathers most of the data from SIM cards. But it produces a simple text file rather than a secure image file. Also, the program has severe compatibility issues: it only works with certain smart card readers and is unstable with modern versions of Windows. The imaging tool described in this paper was designed from the ground up to satisfy five requirements.

- *Stability*: The imaging tool should work on modern operating systems without terminating unexpectedly.

- *Completeness*: The imaging tool should extract all the data from SIM cards without damaging them.

- *Preservation*: Evidence extraction techniques which are likely to damage the devices should be avoided. For example, no attempts should be made to read files that are marked as protected. These files are protected by numeric passwords. Attempting to guess a password too many times can render a SIM card unusable.

- *Compatiblity*: The tool should work with popular operating systems, smart cards and smart card readers.

- *Speed*: Although speed is not a primary concern, the image extraction process should be fast enough not to encumber investigations.

5.1 Communications

The SIM card imaging tool operates over USB ports, using the Personal Computer Smart Card (PC/SC) interface [13]. Although other

methods were employed in the development process, PC/SC ended up being the most stable.

Smart card readers used during development include the GemPC Twin and the GemPC Key, both distributed by GEMPLUS. The GemPC Twin was used for the initial serial development purposes, while the GemPC Key was used for PC/SC USB development.

Table 5. Image file format.

Size	Description
32 bits	Image Version (currently 0)
64 bits	Standard 64-bit Time-Stamp (milliseconds)
32 bits	Number of Files
0+ bits	Files
32 bits	XOR Checksum
128 bits	MD2 Signature
128 bits	MD5 Signature
160 bits	SHA1 Signature

5.2 SIM Card Image File

The SIM card image file is outlined in Table 5. It contains a version number, time-stamp, files, and the image file hashes and checksums. Each file is merely a small header, with entries for the file name, record size, etc., followed by the file contents.

The image file also contains multiple hashes and checksums of the entire image to ensure data integrity. The hashes are computed in a cascading manner, meaning that each hash depends on the previous hashes.

Multiple hashes help maintain the integrity of the data. Collision attacks are possible, if not practical, for many hashes and checksums [10, 16, 17]. However, even if almost every integrity check used in the file format is broken, it is unlikely that an attack can break all the checksums and hash functions at the same time. This fact ensures that the file format will remain intact for a reasonable period of time.

For example, to compromise the XOR checksum at the end of a chained hash, it is necessary to take the original XOR checksum, perform the XOR checksum on the changed file, and append the bit sequence that is the XOR of the two checksums to the end of the file. However, doing this would change every other hash in an unpredictable manner. Therefore, even if an attack were to compromise another hash function or checksum, it is unlikely that it would keep the XOR checksum intact.

5.3 Personal Computer Smart Card Interface

Initially, the imaging tool was developed solely for serial port communication protocols. However, serial port communication is slow, even for small data transfers. In addition, smart card readers operate poorly over serial ports with many stability problems. Furthermore, serial port communication protocols for GEMPLUS smart card readers change with each chipset revision.

To combat these problems, a more stable, generic smart card interface based on the Personal Computer Smart Card (PC/SC) interface was also implemented. PC/SC is an operating system and transmission medium independent interface, allowing programs to exchange Application Protocol Data Units (APDUs) with any smart card reader that has a valid PC/SC driver [13]. The drivers also allow the program to operate at speeds greater than standard serial port speeds by using a USB interface, which is present on practically every modern computer.

The PC/SC interface was integrated into Java code with the help of JPCSC [12], a set of wrapper classes built using the Java Native Interface (JNI). The JPCSC interface is now the standard used in the imaging program, as it enhances reliability, speed and compatibility.

Using the PC/SC interface, APDUs are sent to the smart card to read the contents [5, 6, 14], enumerate the SIM card files and extract the data from them. The extracted data are then stored in an image file.

6. SIM Card Analysis

This section describes the design and implementation of an image file analysis tool that complements the SIM card imaging tool presented in Section 5. The tool analyzes SIM card image files, but cannot work directly with SIM cards. Nevertheless, it is useful because investigators may not have access to a SIM card during an entire investigation. Moreover, repeated reading of a SIM card can stress and possibly damage its electronic components.

6.1 Interactive Graphical User Interface

The analysis tool incorporates an interactive graphical user interface (GUI) designed for investigators. The GUI integrates several plug-ins, which are responsible for displaying the contents of specific files. New files can be examined by the tool by incorporating additional plug-ins, without recompiling the entire GUI.

The GUI also enables investigators to print formatted reports of data found on SIM cards. These reports can be used for manual analysis as well as for courtroom presentations.

6.2 Recognized Files

Common types of files that are of interest to investigators are displayed in simple, descriptive formats. File locations are displayed in a list on the left-hand side of the GUI, while the contents of a particular file are displayed on the right-hand side.

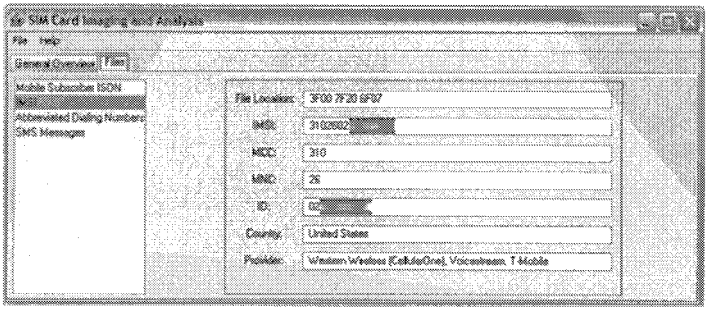


Figure 2. Image file analysis GUI (IMSI)

6.2.1 International Mobile Subscriber Identity. The contents of IMSI files on SIM cards are displayed and analyzed. See Section 4.2.1 for an outline of IMSI file contents. Figure 2 shows the GUI displaying an IMSI file. A portion of the IMSI has been censored so as not to reveal any sensitive information.

6.2.2 Mobile Subscriber ISDN. The MSISDN contains the phone number(s) associated with a SIM card. MSISDNs found on a SIM card are displayed in a standard format. For example, U.S. MSISDNs are presented in the form: `(xxx) yyy-zzzz`. Figure 3 shows the GUI displaying the MSISDN file.

6.2.3 Contact Lists. Contact lists are displayed as accurately as possible. The primary issue with contact lists is that manufacturers often append extra bits of information to the ends of names to provide grouping information, e.g., to assign a specific ring tone for a caller. However, the names and phone numbers should still be identifiable.

Contact lists are found in several SIM files. The most common is the Abbreviated Dialing Numbers (ADN) file, located at `3F00 7F10 3F3A`. Figure 4 shows a GUI display of contact information in an ADN file.

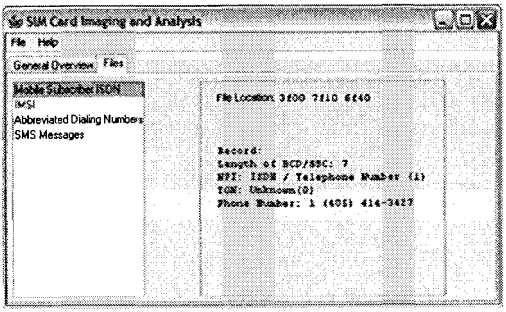


Figure 3. Image file analysis GUI (MSISDN).

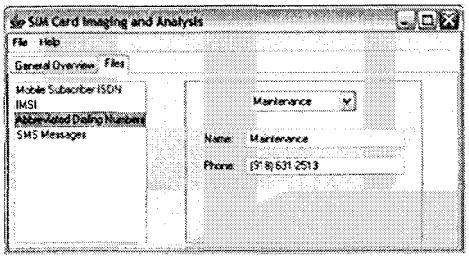


Figure 4. Image file analysis GUI (ADN).

6.2.4 Short Message Service. Short Message Service (SMS), or text messaging, is becoming very popular. The text messages often contain a wealth of information. However, an SMS file on a cellular phone will normally only store incoming messages. Outgoing messages are stored in the device memory, if at all. Figure 5 shows the GUI displaying an SMS file.

6.2.5 Unknown Files. SIM card files that do not have specific plug-ins for recognizing them are processed with a generic plug-in, which displays the data in hexadecimal. Bytes in the ASCII printable character range (numbers, letters, punctuation) are displayed as well.

7. Conclusions

The imaging and analysis tools described in this paper enable law enforcement agents to acquire and analyze information from SIM cards in GSM cellular phones. These tools do not allow investigators to acquire and examine the digital information stored in cellular phone handsets. Nevertheless, information stored in SIM cards is valuable and can be processed without compromising its integrity.

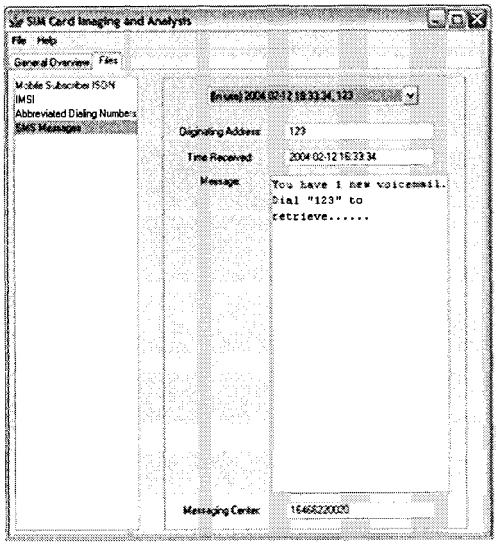


Figure 5. Image file analysis GUI (SMS).

The use of SIM cards as primary storage in cellular phones is waning as flash memory technology (e.g., Secure Digital and CompactFlash) becomes inexpensive. Nevertheless, SIM cards will not disappear anytime soon. The number of GSM subscribers continues to grow. Moreover, Universal SIM (USIM) cards are being deployed with the Universal Mobile Telecommunications System (UMTS), the third generation (3G) successor to GSM. USIM cards are similar to GSM SIM cards and share their physical and logical characteristics. This will permit the implemented tools to be easily extended to USIM card imaging and analysis.

References

[1] B. Carrier, Defining digital forensic examination and analysis tools, *International Journal of Digital Evidence*, vol. 1(4), 2003.

[2] European Telecommunications Standards Institute, 3GPP TS 24.011 v6.0.0 (2003-09), 3rd Generation Partnership Project, Technical Specification Group Core Network; Point-to-Point (PP) Short Message Service (SMS) Support on Mobile Radio Interface (Release 6), 2003.

[3] European Telecommunications Standards Institute, 3GPP TS 23.039 v6.1.0 (2004-09), 3rd Generation Partnership Project, Technical Specification Group Terminals, Alphabets and Language-Specific Information (Release 6), 2004.

[4] European Telecommunications Standards Institute, 3GPP TS 23.040 v5.5.0 (2004-09), 3rd Generation Partnership Project, Technical Specification Group Terminals, Technical Realization of Short Message Service (SMS), (Release 6), 2004.

[5] European Telecommunications Standards Institute, ETSI TS 102 221 v7.0.0 (2004-12), Smart Cards, UICC-Terminal Interface, Physical and Logical Characteristics (Release 7), 2004.

[6] European Telecommunications Standards Institute, ETSI TS 151 011 v4.11.0 (2004-03), Digital Cellular Telecommunications System (Phase 2+), Specification of the Subscriber Identity Module-Mobile Equipment (SIM-ME) Interface (3GPP TS 51.011 Version 4.11.0 Release 4), 2004.

[7] P. Hawkins, Macintosh forensics analysis using OS X (www.sans.org/rr/papers/34/269.pdf), 2002.

[8] International Telecommunications Union, List of E.212 mobile country codes (www.numberingplans.com).

[9] M. Karim and M. Sarraf, *W-CDMA and cdma2000 for 3G Mobile Networks*, McGraw-Hill Professional, Martinsburg, Virginia, 2002.

[10] V. Klíma, Finding MD5 collisions – A toy for a notebook (eprint.iacr.org/2005/075), 2005.

[11] K. Mandia and C. Prosise, *Incident Response: Investigating Computer Crime*, McGraw-Hill, Berkeley, California, 2001.

[12] M. Oestreicher, JPCSC: JNI-wrapper for PCSC (www.zurich.ibm.com/jcop/download/tools/data/jpcsc-0.7.txt).

[13] PC/SC Working Group, Interoperability specification for ICCs and personal computer systems (www.pcscworkgroup.com), 2004.

[14] W. Rankl and W. Effing, *Smart Card Handbook*, John Wiley, Chippenham Wiltshire, Great Britain, 2003.

[15] N. Sollenberger, N. Seshadri and R. Cox, The evolution of IS-136 TDMA for third-generation wireless services, *IEEE Personal Communications Magazine*, vol. 6(3), pp. 8-18, 1999.

[16] X. Wang, Y. Yin and H. Yu, Collision search attacks on SHA1 (theory.csail.mit.edu/~yiqun/shanote.pdf), 2005.

[17] X. Wang and H. Yu, How to break MD5 and other hash functions (www.infosec.sdu.edu.cn/paper/md5-attack.pdf), 2005.

Chapter 18

EXTRACTING CONCEALED DATA FROM BIOS CHIPS

P. Gershteyn, M. Davis, G. Manes and S. Shenoi

Abstract The practice of digital forensics demands thorough, meticulous examinations of all data storage media seized in investigations. However, BIOS chips and other firmware are largely overlooked in forensic investigations. No forensically sound procedures exist for imaging BIOS chips and no tools are available specifically for analyzing BIOS image files. Yet, significant amounts of data may be stored on BIOS chips without hindering machine performance.

This paper describes robust techniques for concealing data in BIOS freespace, BIOS modules, and throughout a BIOS chip. Also, it discusses how flashing utilities and traditional digital forensic tools can be used to detect and recover concealed data.

Keywords: BIOS chips, firmware, data concealment, evidence acquisition

1. Introduction

The Basic Input/Output System (BIOS) of a computer is an interface that enables its hardware and software to interact with each other [7, 16, 18]. BIOS chips provide diagnostics and utilities necessary for loading operating systems. No computer – from the smallest embedded device to the largest supercomputer – can function without a BIOS.

BIOS chips typically contain 128 to 512K of flash memory, which can be used to conceal data. A BIOS writing technique was exploited by the 1998 Win95/CIH virus that wiped out hard drives. Computer game enthusiasts often use BIOS editing to "mod" computers with personalized graphics [3]. We were able to store 40 pages of *The Jolly Roger's Cookbook* [11] on a functioning BIOS chip. Criminals can adopt similar techniques to conceal information: drug contacts, financial records, digital photographs, or cryptographic keys that encrypt child pornography

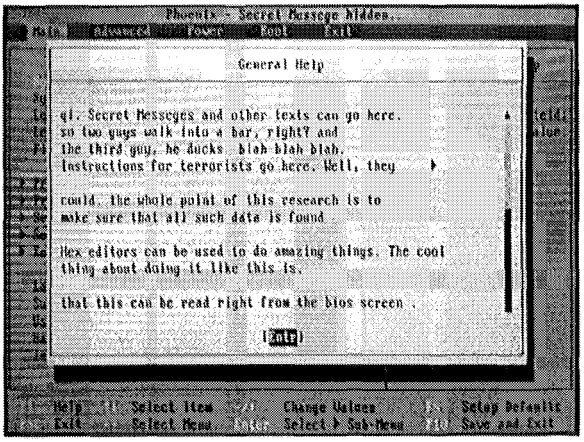


Figure 1. BIOS setup screen displaying hidden data.

stored on hard drives. However, BIOS chips are largely overlooked in forensic investigations. No forensically sound procedures exist for imaging BIOS chips and no tools are available specifically for analyzing BIOS image files.

This paper describes robust techniques for concealing data in BIOS freespace, BIOS modules, and throughout a BIOS chip. Also, it discusses how flashing utilities and traditional digital forensic tools can be used to detect and recover concealed data.

2. BIOS Overview

The Basic Input/Output System (BIOS) of a computer is an interface that enables its hardware and software to interact with each other [7, 16, 18]. Typically located on the motherboard, the BIOS contains software necessary for the computer to start, including instructions for performing a Power-On Self-Test (POST) and reading hard drive boot sectors [2, 18]. BIOS chips also offer basic diagnostics utilities and provide low-level routines that operating systems may use for communicating with hardware. BIOS configurations are stored on a CMOS chip and powered by a small lithium or nickel-cadmium battery that allows the CMOS to store data for several years. Modern BIOS chips use flash memory that enables them to be modified, updated and erased. BIOS chips on most modern computers have storage capacities between 128 and 512K.

Executable code within a BIOS is typically active only during the boot process and until operating system hardware drivers are loaded into memory [5]. From this point on, operating system commands are used to interact with hardware devices and the BIOS maintains only limited

control over low-level hardware features such as power management and certain software interrupts.

BIOS software is modular, enabling it to perform separate functions at startup depending on the computer's hardware configuration. The software consists of two main parts: compressed modules and an area called the "bootblock," which is responsible for decompressing and executing the modules. Hardware that requires initialization can be recognized and loaded by the BIOS. The BIOS checks reserved memory locations for specific byte sequences and headers that indicate the location and size of the hardware initialization code. Modern BIOSs check memory from 0x0C0000 to 0x0EFFFF at 2K boundaries, sidestepping memory ranges used by video hardware and the system BIOS itself. This behavior provides for plug-and-play functionality of hardware devices at the BIOS level [7].

2.1 BIOS Boot Process

When a computer is powered on, the processor first accesses a predetermined area of the system BIOS ROM to access the BIOS boot program. This location is at memory offset 0xFFFF0, sixteen bytes below the top of real mode memory [12]. Real mode is the default operating mode for modern Intel-architecture processors, allowing the processor to mimic the 8088 chip. Offset 0xFFFF0 is not located in RAM, but actually on the BIOS ROM chip. This location contains a jump instruction to a memory offset that contains the BIOS startup code, typically 0xFE05B.

At this stage of the process, most flash BIOS chips copy themselves to RAM in a process known as "shadowing," which allows faster access to the BIOS code while also decompressing and defragmenting noncontiguous BIOS code. The BIOS code then performs the Power-On Self-Test (POST). POST checks if all necessary hardware is present and functioning normally. Since the video card is not active at this point, POST uses beep codes to alert the user to fatal errors encountered during the boot process [2].

Next, the BIOS runs the video card's built-in BIOS program, which is normally found at location 0xC000. The video card's BIOS initializes the video card and displays the confirmation on the screen, providing the first visual indication that the computer is booting. The system BIOS then looks for and executes all other BIOS-enabled devices, displays the startup screen, and conducts more system tests, including a memory count. All fatal errors encountered by the BIOS at this point are displayed on the screen.

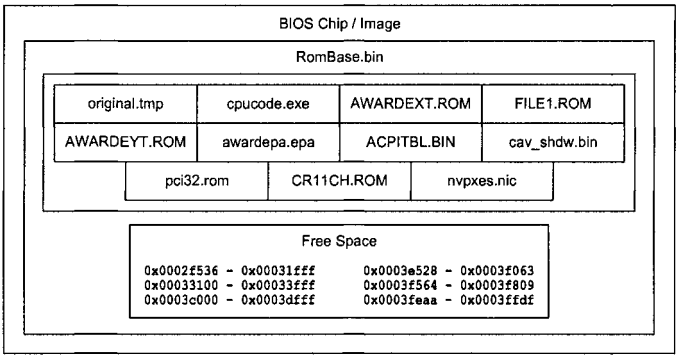


Figure 2. Schematic diagram of BIOS storage.

The BIOS then configures system hardware, determines memory timing and dynamically sets hardware parameters [12]. The BIOS stores hardware parameters in a 256 byte portion of reserved RAM between locations 0x000400 and 0x0004FF, which allows the operating system and other programs to reference hardware information. Once the BIOS completes its inventory of computer hardware, it displays a summary and begins searching for a boot device. After the BIOS finds a device containing boot information it executes code from the first sector of that device.

2.2 BIOS Storage

This work focuses on an ASUS A7N266-VM motherboard with a socketed BIOS chip [1]. The BIOS version is the original ASUS A7N266-VM ACPI BIOS Rev. 1004/AA with build date 08/23/02.

Figure 2 shows the storage structure of the ASUS A7N266-VM BIOS. It consists of eleven modules stored at the start of the file, followed by BIOS data interspersed with freespace.

Data may be stored at several sites on a BIOS chip's flash memory. Depending on the location and quantity of data stored, the BIOS could remain functional or it could become corrupted, rendering the computer inoperable. BIOS chips are designed to be expandable. Consequently, they have large portions of freespace that can be overwritten with data without adversely affecting the operation of the chip. Figure 3 shows a schematic diagram of a BIOS chip with data hidden in its freespace.

Data may also be stored within BIOS modules. Modules often contain text strings that are displayed as messages, e.g., obscure error messages and hardware data. These text strings can be overwritten with data without affecting the operation of the chip. Figure 4 shows a BIOS chip

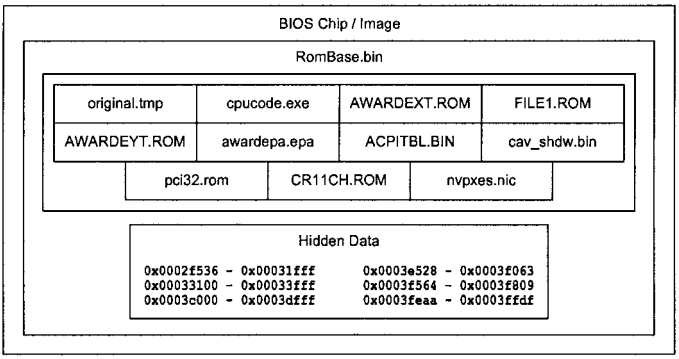


Figure 3. BIOS with hidden data in freespace.

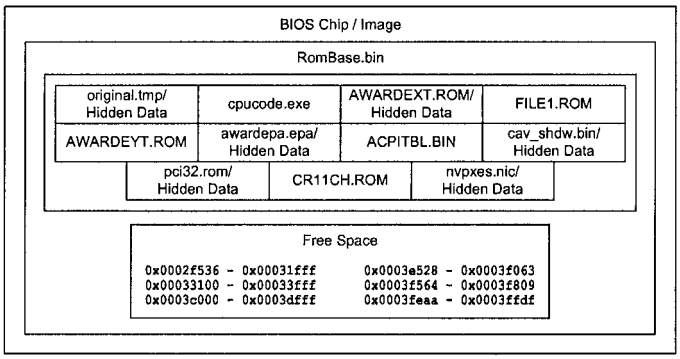


Figure 4. BIOS with hidden data in modules.

with hidden data in six modules. A procedure for concealing data in two of these modules is described in Section 3.2.

If BIOS functionality is not a concern, the entire BIOS chip memory (256K in the case of ASUS) can be used for data storage. However, this makes the recovery of the data problematic, as the computer cannot be booted when the BIOS has been corrupted.

2.3 BIOS Flashing

Computer manufacturers often release updated BIOS software (images or ROMs) for their motherboards [6]. A BIOS is upgraded using a "flashing" program, which erases the chip and replaces its software with the new BIOS image. Sometimes, manufacturers package a flashing program and BIOS image as a single executable file. Alternatively, a third party flashing program (Uniflash [17]), which is compatible with most motherboards and BIOSs can be used to upgrade a BIOS. These flashing programs can be used to conceal data on BIOS chips.

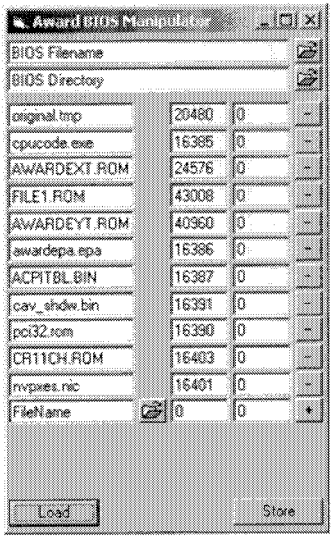


Figure 5. AwardMod screen during extraction of BIOS modules.

Most flashing programs run from the command prompt and require the computer to be running in the DOS mode with no other programs, drivers or services running. Therefore, an MS-DOS boot disk must be modified to create an environment for running a flashing program. Appropriate boot disks, e.g., Caldera Dr-DOS [4], may also be downloaded from the Internet. Newer motherboards now support BIOS flashing from Windows using special software; this makes it possible to quickly read and write BIOS chips.

A BIOS utility, e.g., AwardMod [8], can be used to extract, delete and add modules to a BIOS image. Figure 5 shows an AwardMod screen during the process of extracting ASUS BIOS modules. Hex editors may also be used to read and modify BIOS modules, except for those containing graphics, e.g., the BIOS boot logo, which is encoded in EPA format. A separate program, such as EPACoder [15], facilitates the editing process by converting between EPA and bitmap graphics. Figure 6 shows EPACoder being used to replace the standard BIOS logo with a skull and crossbones.

Editing BIOS modules with AwardMod can corrupt the chip. To recover from this failed flashing attempt, it is necessary to boot the computer in order to re-flash the BIOS. Since a computer with a corrupt BIOS will not boot, the "hotflashing" technique [9] must be used. Hotflashing involves replacing the corrupt BIOS chip with a working chip, booting the computer to a state that allows flashing, and then switching

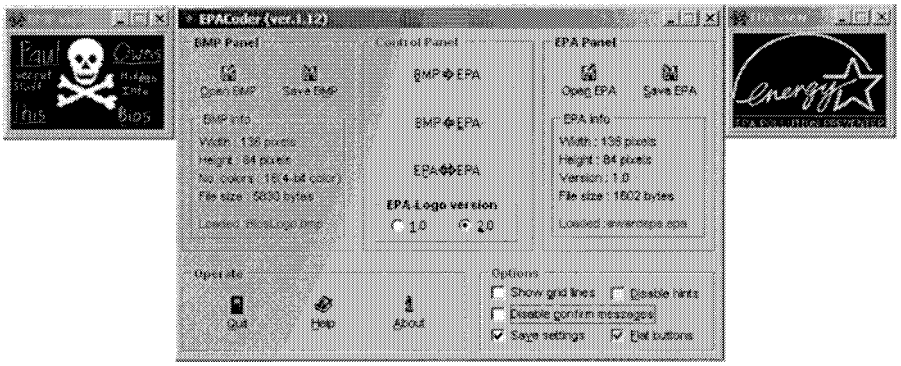


Figure 6. EPACoder screen during bitmap to BIOS image conversion.

the working chip with the corrupt chip while the computer is running. This permits the corrupt chip to be re-flashed.

Special hardware tools, e.g., BIOS Savior [10], simplify the hotflashing process. BIOS Savior interfaces the motherboard and BIOS chip, and provides a backup BIOS chip. A switch allows the user to choose between the original BIOS and the BIOS Savior backup chip. Thus, the user can hotswap BIOS chips with the flip of a switch rather than having to physically remove and insert chips into a running computer.

3. Data Concealment

This section describes techniques for concealing data in: (i) BIOS freespace, (ii) BIOS modules, and (iii) throughout the BIOS chip. The first two techniques produce a usable BIOS chip with hidden data. The third technique can hide a substantial amount of data, but it renders the chip unusable. Nevertheless, the hidden data can be extracted using special techniques (see Section 4).

The BIOS Savior tool [10] is used for hotflashing [9]. Caldera Dr-DOS [4] is used to boot the computer into a state where flashing the BIOS chip is possible. An ASUS flashing program (`aflash.exe` [1]) is used to read and write to the BIOS chip. AwardMod [8] is used to extract and replace BIOS modules. A hex editor (Hex Workshop) is used to edit BIOS data. EPACoder [15] is used to convert graphical images to a usable format. A separate workstation is used to manage the process of flashing the BIOS.

3.1 Overwriting BIOS Freespace

Because BIOS software is designed to be upgradeable, BIOS chips typically have significant amounts of freespace. The following procedure

describes how data may be hidden in BIOS freespace without affecting its operation. Note, however, that prior research is necessary to verify which blocks of freespace can be used without affecting a BIOS chip.

BIOS Freespace Overwriting Procedure

1 Procure an MS-DOS compatible boot disk approved for BIOS flashing (e.g., Caldera Dr-DOS). The disk should not execute any terminate and stay resident (TSR) programs.

2 Copy the ASUS flashing program (`aflash.exe`) to the boot disk.

3 Boot the ASUS machine using the boot disk. This may require altering the drive boot order in the CMOS settings. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk and save the file on the boot disk as `asback.bin`.

4 Place the boot disk in the workstation and copy `asback.bin` to the hard drive.

5 Find all 8 blocks of null characters in `asback.bin`. Null blocks are long strings of either `0`s or `F`s. Since these blocks represent free space, data may be written to them without corrupting the BIOS. The null blocks are present at the following locations:

```
Block 1:  FFFFs at 0x0002F536--0x00031FFF
Block 2:  0000s at 0x00032A26--0x00032FFD
Block 3:  0000s at 0x00033100--0x00033FFF
Block 4:  FFFFs at 0x0003B6A0--0x0003BFFF
Block 5:  0000s at 0x0003C000--0x0003DFFF
Block 6:  0000s at 0x0003E528--0x0003F063
Block 7:  0000s at 0x0003F564--0x0003F809
Block 8:  0000s at 0x0003FEAA--0x0003FFDF
```

Note that editing any part of Block 2 corrupts the BIOS. Also, while editing Block 4 will not corrupt the BIOS, the stored data cannot be recovered. The remaining blocks permit both data storage and retrieval.

6 Select a file (`evidence.rar`) to be hidden that can fit within the null blocks, which in this case is 26,850 bytes. Compression may be used to store more data than would otherwise be possible. Multiple files may also be stored as long as they do not exceed a total of 26,850 bytes.

7 Write `evidence.rar` across the empty blocks. Blocks that are not filled with data should be padded with zeros. A file that is too large for a block can be split using a hex editor and written to multiple blocks.

8 After the null bytes of `asback.bin` are overwritten by the data in `evidence.rar`, save `asback.bin` and rename it `asedited.bin`.

9 To complete the process of hiding `evidence.rar`, copy `asedited.bin` to the boot disk, boot the ASUS using the boot disk, and flash the BIOS. This is done by typing `aflash /boot /auto asedited.bin` at the command prompt.

10 Restart the computer to verify that it still functions properly.

3.2 Editing BIOS Modules

Significant amounts of data may be hidden within BIOS modules without affecting the operation of the chip. Modules often contain text strings that are displayed as messages (e.g., obscure error messages and hardware data); these text strings can be overwritten with data. Likewise, modules contain considerable amounts of freespace that can also be used to hide data.

The procedure for concealing data in two modules, `awardepa.epa` and `awardext.rom`, is described below. The module `awardepa.epa` stores the BIOS boot logo; `awardext.rom` contains text that is displayed at the BIOS setup screen. Data hidden in these modules can be read from the screen without having to image the BIOS. Note that the boot logo is stored as `awardepa.epa` in one of two formats, EPA 1 and EPA 2. EPA 1 is a limited format: a picture is divided into cells, each cell limited to two colors. Despite this restriction, it is possible to store information in this graphic. The ASUS BIOS only supports EPA 1.

BIOS Module Editing Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program (`aflash.exe`) to the boot disk.

3 Boot the ASUS machine using the boot disk. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk as `asback.bin`. Place the boot disk in the workstation and copy `asback.bin` to the hard drive.

4 Copy `asback.bin` to the workstation from the boot disk and run AwardMod. AwardMod can be used to load and store a binary BIOS file or a directory containing the extracted modules from the binary BIOS file. Use AwardMod to extract the modules in `asback.bin` and store them in a directory.

5 Use a hex editor to edit the extracted module `awardext.rom`. The strings in this file are the same strings that are visible on the BIOS setup screen. Since the help text is scrollable and relatively long, this is the best place to write data. Save the edited version of `awardext.rom` as `zawardext.rom`.

6 Use EPACoder to convert `awardepa.epa` to a bitmap. Edit the bitmap using a graphics program and convert it back to the EPA 1 format using EPACoder.

7 Use AwardMod to open `asback.bin`. Copy the module number associated with `awardext.rom`. Then, delete `awardext.rom`. Next, enter the location of `zawardext.rom` in the new module (file name) prompt in AwardMod. Also, paste `awardext.rom`'s module number into the box associated with the new module, and add this module to the BIOS.

8 Similarly, delete `awardepa.epa` and add `zawardepa.epa`, making sure that `zawardepa.epa` has the same module number as `awardepa.epa`.

9 Enter the new file name `asedited.bin` and its location in the BIOS file name prompt of AwardMod to write this file on the hard drive. Flash this file to the ASUS BIOS chip using the boot disk and the command `aflash.exe /boot /auto asedited.bin`.

10 Restart the computer to verify that it still functions properly. Also verify that the changes made to the modules are reflected in the BIOS startup and setup screens.

3.3 Overwriting the Entire BIOS

As discussed earlier, the entire BIOS chip memory (256K in the case of ASUS) can be used to hide data. However, this makes the recovery of the data problematic, as the computer cannot be booted with a corrupted BIOS. Hotflashing is one solution. Alternatively, the motherboard must have two BIOS chips or a BIOS backup device, e.g., BIOS Savior, must be used to recover the hidden data. The procedure described below makes use of BIOS Savior. The user can flip a switch on the computer to choose whether the original chip or the BIOS Savior chip should be used.

Entire BIOS Overwriting Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program (`aflash.exe`) to the boot disk.

3 Create a RAR archive with the files to be hidden. The files should not be compressed to ensure that the size of the RAR archive is predictable.

4 When the RAR archive is close to (but less than) 262,144 bytes save and close the archive. Then, open it in a hex editor and pad with zeros at the end until the total size of the RAR archive is exactly 262,144 bytes. Name the RAR archive `evidence.rar`.

5 Copy `evidence.rar` to the boot floppy. Boot the ASUS with the boot floppy and flash `evidence.rar` to the BIOS chip using the command `aflash /boot /auto evidence.rar`. Note that the ASUS will no longer be able to boot with its BIOS chip.

4. Concealed Data Recovery

This section describes procedures for detecting and extracting data that has been hidden using the procedures described in Section 3. Data hidden in the freespace cannot be detected without imaging the BIOS chip. However, it may be possible to detect if data is hidden within BIOS modules before imaging the BIOS, depending on which modules contain hidden data. It is always possible to detect whether or not the

entire BIOS chip has been overwritten with data simply by turning on the computer.

The following procedure should be followed for investigating a seized computer that may have data hidden in its BIOS chip.

Initial Investigative Procedure

1 Turn on the seized computer after its hard drives have been removed. If the computer does not boot, it is possible that the entire BIOS chip has been overwritten with data.

2 Examine the BIOS startup and setup screens for any unusual text or graphics. The existence of anomalies indicates that the BIOS modules have been edited. Note, however, that the absence of anomalies does not guarantee that the BIOS modules are free of hidden data.

3 Search the seized storage media for BIOS modification tools and flashing programs. The presence of such software may provide clues with regard to the type of data hidden in the BIOS as well as the technique used.

The following three subsections describe procedures for detecting and extracting hidden data from various locations in a BIOS chip.

4.1 Recovering Data from BIOS Freespace

The following procedure should be used if the investigator suspects that data is hidden on the BIOS chip, although there is no visible evidence of it on the BIOS startup and setup screens. Note that BIOS modules should also be checked for hidden data.

BIOS Module Freespace Recovery Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program `aflash.exe` to the boot disk.

3 Boot the ASUS machine using the boot disk. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk as `asback.bin`.

4 Place the boot disk in the workstation and copy `asback.bin` to the hard drive.

5 Use forensic utilities (e.g., Foremost, Encase, Forensic Tool Kit, ILook [13, 14]) to examine the BIOS image for file headers and regular expressions, and preserve all data of interest.

6 If the hidden data cannot be found using the forensic utilities, use a hex editor to compare the seized BIOS image with a clean copy of the BIOS image from the motherboard manufacturer's website. This comparison assists in locating hidden data.

7 If a clean copy of the BIOS image is not available, examine the seized BIOS's image with a hex editor and look for suspicious text strings.

8 Use forensically sound procedures to copy and preserve all data of interest.

4.2 Recovering Data from BIOS Modules

If hidden data is found upon examining all the BIOS setup screens, the BIOS modules must be processed to recover the evidence. Even if no hidden data is found within the BIOS startup and setup screens, it is still possible for data to be hidden within the modules. The following procedure should be performed along with the procedure to recover hidden data from BIOS freespace (Section 4.1).

BIOS Module Data Recovery Procedure

1 Procure an MS-DOS compatible boot disk (Caldera Dr-DOS) approved for BIOS flashing.

2 Copy the ASUS flashing program `aflash.exe` to the boot disk.

3 Boot the ASUS machine using the boot disk. After Caldera Dr-DOS has booted, execute `aflash.exe`. Backup the original BIOS to the floppy disk as `asback.bin`.

4 Place the boot disk into the workstation and copy `asback.bin` to the hard drive.

5 Use AwardMod to extract all modules from `asback.bin`.

6 Use forensic utilities (e.g., Foremost, Encase, Forensic Tool Kit, ILook [13, 14]) to examine the BIOS modules for file headers and regular expressions, and preserve all data of interest.

7 If the hidden data cannot be found using the forensic utilities, use a hex editor to compare the seized BIOS's modules with those in a clean copy of the BIOS image from the motherboard manufacturer's website. This comparison assists in locating hidden data.

8 If a clean copy of the BIOS image is not available, examine the seized BIOS's modules with a hex editor and look for suspicious text strings.

9 Use forensically sound procedures to copy and preserve all data of interest.

4.3 Recovering Data from Entire BIOS Chip

Recovering data from a BIOS chip that has been overwritten completely requires the chip to be physically removed. The hotflashing technique or a BIOS Savior may be used to image the seized BIOS chip. Alternatively, a chip programmer can be used.

Data of any type may be hidden on a BIOS chip. Therefore, it is recommended that forensic tools be used to conduct extensive examinations of the BIOS image. Careful manual examination of the hex code must also be performed. Forensically sound procedures must be used to copy and preserve all data of interest.

5. Conclusions

Modern BIOS chips can hold substantial amounts of hidden data without affecting their performance. This paper shows how data may be hidden in BIOS freespace, BIOS modules, and throughout a BIOS chip. Also, it presents forensically sound techniques for detecting and recovering concealed data. The work is intended to raise awareness about the ability of malicious individuals to store secret information on BIOS chips and other firmware. Moreover, it should stimulate new research in the area of firmware forensics.

References

[1] ASUS, A7N266-VM/AA motherboard support (support.asus.com), 2003.

[2] BIOS Central (www.bioscentral.com).

[3] BIOSMods (www.biosmods.com).

[4] Bootdisk.com (bootdisk.com).

[5] P. Croucher, *The BIOS Companion*, Electrocution Technical Publishers, Calgary, Alberta, Canada, 1998.

[6] W. Gatliff, Implementing downloadable firmware with flash memory, in *The Firmware Handbook*, J. Ganssle (Ed.), Elsevier, Burlington, Massachusetts, pp. 285-297, 2004.

[7] Gen-X-PC, BIOS info (www.gen-x-pc.com/BIOS_info.htm).

[8] J. Hill, AwardMod (sourceforge.net/projects/awardmod/), 2002.

[9] K. Hindistan, BIOS flashing and hotflashing (www.onlamp.com/pub/a/onlamp/2004/03/11/bios_hotflash.html), 2004.

[10] IOSS, RD1 BIOS Savior (www.ioss.com.tw), 2000.

[11] Jolly Roger, *The Jolly Roger's Cookbook* (www.textfiles.com), 1990.

[12] C. Kozierok, System BIOS (www.pcguide.com), 2001.

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

[14] G. Mohay, A. Anderson, B. Collie, O. de Vel and R. McKemmish, *Computer and Intrusion Forensics*, Artech, Norwood, Massachusetts, 2003.

[15] S. Nikolayev and A. Prokopiuk, EPACoder (shareware.pcmag.com /product.php[id]38610[cid]301[SiteID]pcmag), 2000.

[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] A. Wong, *Breaking Through the BIOS Barrier: The Definitive BIOS Optimization Guide for PCs*, Prentice Hall, Indianapolis, Indiana, 2004.

V

LINUX AND FILE SYSTEM FORENSICS

Chapter 19

RECOVERING DIGITAL EVIDENCE FROM LINUX SYSTEMS

Philip Craiger

Abstract As Linux-kernel-based operating systems proliferate there will be an inevitable increase in Linux systems that law enforcement agents must process in criminal investigations. The skills and expertise required to recover evidence from Microsoft-Windows-based systems do not necessarily translate to Linux systems. This paper discusses digital forensic procedures for recovering evidence from Linux systems. In particular, it presents methods for identifying and recovering deleted files from disk and volatile memory, identifying notable and Trojan files, finding hidden files, and finding files with renamed extensions. All the procedures are accomplished using Linux command line utilities and require no special or commercial tools.

Keywords: Digital evidence, Linux system forensics

1. Introduction

Linux systems will be increasingly encountered at crime scenes as Linux increases in popularity, particularly as the OS of choice for servers. The skills and expertise required to recover evidence from a Microsoft-Windows-based system, however, do not necessarily translate to the same tasks on a Linux system. For instance, the Microsoft NTFS, FAT, and Linux EXT2/3 file systems work differently enough that understanding one tells little about how the other functions. In this paper we demonstrate digital forensics procedures for Linux systems using Linux command line utilities. The ability to gather evidence from a running system is particularly important as evidence in RAM may be lost if a forensics first responder does not prioritize the collection of live evidence.

The forensic procedures discussed include methods for identifying and recovering deleted files from RAM and magnetic media, identifying no-

tables files and Trojans, and finding hidden files and renamed files (files with renamed extensions.

We begin by describing recovering deleted files from RAM on a live (running) Linux system. Because Linux systems are commonly employed as servers, most of the demonstrations are directed toward activities and techniques that intruders are known to use after breaking into a system.

2. Recovering Files from RAM

A deleted file whose contents have been overwritten on disk may still be recovered. To illustrate the forensic technique, say an intruder may execute a program and then delete it from disk to hide its existence. This happens, for example, when an intruder installs a backdoor on the victim system by executing the `netcat` utility, then deleting the utility from the disk. As long as the program remains as a running process in memory, the original executable can be recovered. The file is recoverable because the Linux kernel uses a pseudo file system to track the general state of the system, including running processes, mounted file systems, kernel information, and several hundred other pieces of critical system information [6]. This information is kept in virtual memory and is accessible through the /proc directory. The (partial) listing below shows the contents of the /proc directory on a running Linux system. Each of the numbers below corresponds a processor ID, and is a directory that contains information on the running process.

```
# ls /proc
1 4 4513 4703 4777 execdomains mdstat swaps
1693 40 4592 4705 acpi fb meminfo sys
2 4045 4593 4706 asound filesystems misc
2375 41 4594 4708 buddyinfo fs mm sysvipc
2429 4163 4595 4709 bus ide modules tty
2497 4166 4596 4712 cmdline interrupts mounts uptime
2764 4186 4597 4713 config.gz iomem mtrr version
29 42 4620 4715 cpufreq ioports net vmstat
```

To illustrate the recovery of a deleted file, say that an intruder has downloaded a password cracker and is attempting to crack system passwords – a very common goal for an intruder. The intruder runs the `john` (www.openwall.com) password cracker with a list of passwords in a file called `pass`. The intruder subsequently deletes both the executable and the text file containing the passwords, the executable remains running in memory until the process is killed. The ps command displays running processes. The listing below shows the executable "john" has been called with the "pass" file at 10:10AM, has been running for 22 seconds, and is owned by root.

```
# ps aux | grep john
root 5288 97.9 0.0 1716 616 pts/2 R+ 10:10 0:22 ./john pass
```

According to the listing above the executable process ID (PID) is 5288. The directory /proc/5288 will contain information regarding the running process, as displayed in the (partial) listing below.

```
# ls -al /proc/5288
total 0
dr-xr-xr-x 3 root root 0 Jan 17 10:11 .
dr-xr-xr-x 108 root root 0 Jan 17 04:00 ..
-r--r--r-- 1 root root 0 Jan 17 10:11 cmdline
lrwxrwxrwx 1 root root 0 Jan 17 10:12 cwd -> /j
-r-------- 1 root root 0 Jan 17 10:12 environ
lrwxrwxrwx 1 root root 0 Jan 17 10:12 exe -> /j/john (deleted)
lrwxrwxrwx 1 root root 0 Jan 17 10:12 root -> /
-r--r--r-- 1 root root 0 Jan 17 10:11 stat
-r--r--r-- 1 root root 0 Jan 17 10:12 statm
dr-xr-xr-x 3 root root 0 Jan 17 10:12 task
```

Directory /proc/5288 contains several files and directories, the most important of which is `exe` which is a symbolic link (note the `l` in the very first column of the permissions) to the running password cracker. The operating system (helpfully) displays a note indicating that the file was deleted from disk. Nevertheless, we can recover the file by copying the `exe` from the directory to a separate directory.

```
# cp /proc/5288/exe ./john.recovered
# md5sum ./john.recovered ./john.original
83219704ded6cd9a534baf7320aebb7b ./john.recovered
83219704ded6cd9a534baf7320aebb7b ./john.original
```

In the example above we copied `exe` from /proc/5288 to another directory, and then compared the MD5 hash of the executable with a hash of a known copy of John. We see the hashes are the same, indicating that we successfully recovered the file. This method of file recovery works for any type of file as long the process remains in memory.

3. Recovering Files by Type

We can manually recover a file by searching unallocated space for the file header, which is located at the beginning of a file. For instance, say we know that an intruder deleted a directory containing several hundred bitmap graphics. We can search through unallocated space for a sector beginning with BM, the signature for a bitmap graphic. When found we can manually recover the file using the Linux `dd` command. The success of this procedure is assumes that: (i) we can identify header information,

(ii) the file has not been overwritten, and (iii) the file is not fragmented. If the file is fragmented, we will only be able to recover part of the file, as we will be unable to identify the blocks that previously comprised the file. In this demonstration we have an image (or unmounted partition) that contains several deleted *.jpg files. First we must identify the first sector of each JPG file, which we do by searching for the text JFIF commonly found in a JPG file. The list below shows a starting sector for a deleted JPG file. (Note: The only part of the header required for a *.jpg file are the first three bytes: `ffd8ffe0`. In experiments we found that deleting the JFIF in the header does not prevent applications from accurately identifying the type of file, although removing any of the first three bytes does.)

```
0004200:  ffd8 ffe0 0010 4a46 4946 0001 0200 0064
```

The listing above shows that the file starts at 0x4200 (hex). We convert this to decimal, 16,896, and divide by 512 (the number of bytes in sector) resulting in 33, which is the starting sector number of the file, i.e., from the beginning of the image. Under many circumstances we will not know the exact size of a deleted file, requiring us to make an educated guess as to its size. If we guess too low and under recover the file, viewing the file in its application will show that too few sectors were recovered, and the file will not appear complete. If we recover too many sectors, we have an over recovery. In our experience recovering too many sectors does not harm the file. Once we recover the file we can view it in the appropriate application to determine the accuracy of our guess. We use the UNIX/Linux `dd` command to carve the file from the image.

We specify the input file to be our image (`if=image.dd`), and we choose a name for the recovered file (`of=recovered1.jpg`). We must specify the starting sector to begin carving. According to our earlier calculation the image starts at physical sector 33. The default block size in `dd` is 512 bytes, which we will leave as is. Finally we must specify the number of consecutive blocks to recover. In this instance we will guess 30 blocks of size 512 bytes each, so we are recovering files of size 15K.

```
# dd if=image.dd of=recovered1.jpg skip=33 count=30
30+0 records in
30+0 records out
# file recovered1.jpg
recovered1.jpg:  JPEG image data, JFIF standard 1.01
```

We successfully recovered 30 consecutive sectors of the JPG file. The file command shows we recovered the header successfully.

This recovery method can be used with any type of file, as long as the file header information remains intact. The success of this method

depends, again, on the lack of file fragmentation and some luck as to whether any blocks of the file have been reused.

3.1 General File Recovery Procedure

If a file header has been overwritten and the file is primarily text, we can use a more general recovery procedure that only requires that we know some keywords for which to search, and of course, that the file has not been completely overwritten.

For this demonstration we will recover the Linux general log file `/var/log/messages`. This file is often targeted by an intruder as it will contain evidence of the intruder tracks. Novice intruders will delete the entire file, which is clearly evidence to an administrator that an intrusion occurred. In contrast, skilled intruders will surgically remove lines that point to their break-in, keeping the remaining contents. To recover the log file we must identify keywords contained in the file. Ideally we identify keywords that are unique to the file, thus reducing the number of false-positive hits. For this example we are likely to encounter some false-positives because log files are rotated on a frequent basis, so our search is likely to pick up keywords from previous versions of the log file. We unmount the partition that contains the directory /var where messages resides. This is simple if /var is on its own partition:

```
# umount /dev/hda3
```

If /var is on the same partition as the root directory we will need to reboot the system using a Linux bootable CD and perform the procedures from the boot disk [2]. Next use we use grep to search for the keywords on the physical device (unmounted partition). We are using the physical device because we must access unallocated space through the physical device:

```
# grep -ia -f keywords -C 2 /dev/hda3
```

The flag `i` specifies a a case insensitive search. The flag `a` specifies to treat the input (contents of the physical device /dev/hda3) as ASCII text; if we do not then grep will only indicate whether the file contains the keyword or not. The flag `f` specifies that what follows is a text file that contains a list of keywords for which to search. We are essentially conducting a simultaneous search for multiple keywords, which we might use, for example, if we are unsure as to exactly what keywords our deleted file contains. The flag `-C 2` specifies that we want two lines of context two lines before and after a keyword hit. Finally we specify the physical device to search which contained the /var directory. For this

demonstration we assume that the attacker made several unsuccessful attempts to log into the root account – a common occurrence in an intrusion. These unsuccessful login attempts will be noted in the messages log file. The results of our search are displayed below:

```
Dec 18 19:13:09 gheera gdm(pam_unix)[2727]:  authentication failure;
logname= uid=0 euid=0 tty=:0 ruser= rhost= user=schmoopie
Dec 18 19:13:13 gheera gdm-binary[2727]:  Couldnt authenticate user
Dec 18 19:13:16 gheera gdm(pam_unix)[2727]:  session opened for user
schmoopie by (uid=0)
Dec 20 18:33:29 gheera gdm(pam_unix)[2752]:  authentication failure;
logname= uid=0 euid=0 tty=:0 ruser= rhost= user=schmoopie
Dec 21 18:16:55 gheera gdm(pam_unix)[2750]:  authentication failure;
logname= uid=0 euid=0 tty=:0 ruser= rhost= user=schmoopie
Dec 22 17:49:33 gheera gdm(pam_unix)[2756]:  authentication failure;
logname= uid=0 euid=0 tty=:0 ruser= rhost= user=schmoopie
Dec 22 17:49:36 gheera gdm-binary[2756]:  Couldnt authenticate user
Dec 22 17:49:48 gheera gdm(pam_unix)[2756]:  session opened for user
schmoopie by (uid=0)
```

The keywords are in bold. It appears that user *schmoopie* unsuccessfully attempted to log in as root on December 18, 19, 21 and 22. Note the two lines of context both before and after each search hit. In practice we would not want such a limited result: We would rather recover the entire contents of the log file, which we could to do by requesting a much larger value for context, e.g., `-C 100`. Because we do not know a priori how large the file is this will be trial and error effort.

3.2 Recovering Files from EXT2 Disks

A final recovery method assumes that the file system is EXT2, a somewhat common Linux file system (although it is being replaced by more efficient journaling file systems). In this method we can use the system debugger to find and recover the file. For this example, say a recently terminated employee deleted an important file from his directory under /home. (Not an uncommon event for terminated employees.) Say we are informed that the file was a zip archive. We must determine the hard drives geometry, including the number of partitions, how the partitions are formatted, before we begin the file recovery process. The Linux command fdisk -l provides this information:

```
# fdisk -l
Disk /dev/hda:  30.0 GB, 30005821440 bytes
16 heads, 63 sectors/track, 58140 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Device Boot Start End Blocks Id System
/dev/hda1 1 41613 20972826 83 Linux
```

```
/dev/hda2 57147 58140 500976 f W95 Extd (LBA)
/dev/hda3 41613 52020 5245222+ 83 Linux
/dev/hda5 57147 58140 500944+ 82 Linux swap
```

We see that we have a single 30GB IDE hard drive with four partitions. The first partition (/dev/hda1) is a primary partition formatted in EXT2. The second partition (/dev/hda2) is an extended partition containing two logical partitions, one an EXT2 file system (/dev/hda3) and the second a Linux swap file (/dev/hda5). Next we need to know which directories are mounted on which partitions. We run the mount command which displays this information.

```
# mount | column -t
/dev/hda1 on / type ext2 (rw,acl,user_xattr)
proc on /proc type proc (rw)
tmpfs on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/hda3 on /home type ext2 (rw,acl,user_xattr)
/dev/hdc on /media/cdrom type subfs ...
```

The mount command shows us that the /home directory is mounted on /dev/hda3 device. We unmount the /home directory, or remount it read-only so that there is no possibility of overwriting the deleted file. The more quickly this can be done the better; as the file has a good chance of being overwritten the longer the partition remains mounted. To unmount the directory, we issue the command:

```
# umount /home
```

We use the debugger debugfs to open the partition and recover the deleted file. In the debugger we execute the lsdel command to display inode information on all the deleted files on the partition. (An inode is a data structure that holds file metadata. See [1, 4] for more information on inodes.)

```
# debugfs /dev/hda3
debugfs 1.35 (28-Dec-2004)
debugfs: lsdel
Inode Owner Mode Size Blocks Time deleted 272319 0 100755 3383328
828/ 828 Thu Dec 23 23:45:22 2004 1 deleted inodes found.  lines 1-3/3
(END)
```

The `lsdel` command indicates that a file represented by inode number 272319 was deleted on December 23 and was of size 3MB (comprising 828 blocks). Once we have the inode number we can get more detailed information with the stat command:

```
debugfs: stat <272319>
```

```
Inode: 272319 Type: regular Mode: 0755 Flags: 0x0 Generation:
92194859
User: 0 Group: 0 Size: 3383328
File ACL: 0 Directory ACL: 0
Links: 0 Blockcount: 6624
Fragment: Address: 0 Number: 0 Size: 0
ctime: 0x41cb9ee2 -- Thu Dec 23 23:45:22 2004
atime: 0x41cb9d68 -- Thu Dec 23 23:39:04 2004
mtime: 0x41cb9d68 -- Thu Dec 23 23:39:04 2004
dtime: 0x41cb9ee2 -- Thu Dec 23 23:45:22 2004
BLOCKS:
(0-11):582122-582133, (IND):582134, (12-826):582135-582949
TOTAL: 828
```

The stat command provides us with a variety of information, including the modified, accessed, changed, and deleted date and times of the deleted file. (Unlike NTFS and FAT file systems, the Linux EXT2 file system tracks a file deleted date and time.) The stat command also shows us the number of direct, indirect, and doubly indirect blocks under the BLOCKS section. (For a more thorough explanation of the EXT2 file system see [1, 5]). It appears that all blocks are intact, i.e., no blocks have been overwritten, meaning we can recover the entire file. The dump command takes as argument an inode number and a name to call the recovered file:

```
debugfs: dump <272319> hda3.recovered
```

Once we exit the debugger we determine the recovered file type with the file command. The file command uses the header information to determine the type of file.

```
# file hda3.recovered
hda3.recovered: Zip archive data, at least v1.0 to extract
```

Our recovered file is a ZIP archive, as expected. We determine the success of our procedure by comparing the hash of our recovered file with the hash of the original file (which we happen to have for our demonstration here). The hashes match indicating that we successfully recovered the file. (Or when an MD5 does not exist of the original, simply unzipping the file, in is case.)

```
# md5sum original.file.zip hda3.recovered
ed9a6bb2353ca7126c3658cb976a2dad original.file.zip
ed9a6bb2353ca7126c3658cb976a2dad hda3.recovered
```

The success of this procedure depends on a number of critical factors. First is the time interval between when the file is deleted and attempted

recovery. The longer the time between deletion and recovery, the more likely part or the entire file will be overwritten. A second factor is file size. Smaller files (that fit in the direct blocks) have a higher probability of being recovered than larger files that may also require the use of indirect and doubly indirect blocks.

3.3 Identifying Notable Files and Trojans

The two primary goals of intruders are to effectuate a break in, and to remain on the victim system as long as possible. Remaining hidden on the system is usually accomplished by installing a rootkit. A rootkit replaces several important system files with "Trojaned" versions. The Trojaned versions work like the original system files with the exception that they fail to display any traces of the intruder, such as running processes, open files or open sockets. Utilities that are commonly Trojaned include ps (to display system processes), netstat (to display sockets and network connections), and top (display process information sorted by activity), among others.

A simple way to identify Trojaned files is through a hash analysis. A hash analysis compares the one-way cryptographic hashes of "notable" files with hashes of files on the system. If two hashes match it indicates that a file has been replaced with a Trojaned version.

A second method of identifying Trojans is by comparing inode numbers of files within a directory. An inode number that is substantially out-of-sequence with the inode numbers of other files in a directory could be an indication that the file has been replaced.

When a file is saved to the hard drive it is assigned an inode number. Files that are saved in short succession will have inode numbers that are consecutive or nearly so. This is demonstrated below, which displays a (partial) directory listing of the contents of the /bin directory, sorted by inode number (located in the first column).

```
# ls -ali /bin | sort
130091 -rwxr-xr-x 1 root root 59100 Oct 5 11:50 cp
130092 -rwxr-xr-x 1 root root 15516 Oct 5 11:50 unlink
130093 -rwxr-xr-x 1 root root 161380 Oct 11 09:25 tar
130094 -rwxr-xr-x 1 root root 16556 Oct 5 11:50 rmdir
130095 -rwxr-xr-x 1 root root 26912 Oct 5 11:50 ln
130096 -rwxr-xr-x 1 root root 10804 Sep 30 08:49 hostname
130097 -rwxr-xr-x 1 root root 307488 Sep 21 17:26 tcsh
569988 -rwxr-xr-x 1 root root 76633 Jun 29 2004 ps
569990 -rwxr-xr-x 1 root root 92110 Jan 18 2004 netstat
```

The order in which the files were saved to the hard disk is clear as shown by the increasing sequence of inode numbers. It is clear we have an

abnormality with the inode numbers for the ps and netstat commands. A file inode number will change when it is replaced with a different file.

Because the Trojan was installed well after the original file the Trojan inode number will be higher than that of the original file. Thus, a simple method of identifying Trojans is looking for inode numbers that are "outliers" particularly for those files that are likely to be part of a rootkit. As demonstrated above, the ps and netstat have inode numbers that are significantly out-of-sequence with the inode numbers of the other files, indicating the possibility that the original utility was replaced with a Trojan version. This is not a guarantee, unlike the hash analysis above, that the files are known Trojan horses. Regardless, further scrutiny is warranted.

3.4 Identifying Files with Renamed Extensions

A simple means of hiding a file is by renaming the file extension. For instance, changing the file chix.jpg to homework.doc takes a file of questionable content and turns it into a file that appears innocuous. This technique can be particularly effective within Windows because Windows will display an icon that is based on the extension of a file, regardless as to whether a file extension is a true reflection of the file type. As described previously, a file type is reflected in its header (sometimes called signature). A file header is a sign to applications as to how to handle the file. For instance, all modern Microsoft Office files begin with the following 8-byte signatures (in bold):

```
d0cf 11e0 a1b1 1ae1 0000 0000 0000 0000
```

One way find graphic files whose extension has been changed is to combine three GNU utilities: `find`, `file`, and `grep`. The best way to explain the procedure is through a demonstration.

1. Use the `find` command to find all regular files on the hard drive.
2. Pipe the results of this command to the `file` command, which displays the type of file based on header information.
3. Pipe the results of this command to the `grep` command to search for graphical-related keywords.

Below we combine the three utilities to identify all graphical images that have a renamed extension:

```
# find / -type f ! -name' * .jpg - o - name' * .bmp' - o - name' * .png'
-print0 | xargs -0 file | grep -if graphics.files
```

This is simpler to understand if partitioned into steps:

1. The / argument specifies the directory in which to start, here the root directory.

2 The flag `-type f` specifies that we are interested in regular files as opposed to special files such as devices or directories. The find command is recursive by default so it is essentially recursively finding all regular files beginning at the / (root) directory.

3 The exclamation mark (!) modifies the contents within the parenthesis, and indicates that we want to process files whose extension is not *.jpg, or *.png, or *.bmp, or *.tiff.

4 The print0 is a special formatting command that is required to format the output of find for piping to the next command.

5 Pipe the results a list of files whose extension is not *.jpg, *.bmp, etc. – to `xargs -0`, which sends each file name to the `file` command. file evaluates each file signature, returning a description of the type of file.

6 These results are piped to grep to search for the specific keywords that are contained within the graphics.files file. The arguments for grep include `i` for case insensitive search, and the `f` graphics.files, the file containing the list of keywords: PNG, GIF, bitmap, JPEG and image.

Our search identified three files with misleading names and extensions:

```
# find / -type f !
( -name '*.jpg' -o -name '*.bmp' -o -name '*.png'
) -print0 | xargs-0 file | grep -if graphics.files
/var/cache/exec:  JPEG image data, JFIF standard 1.01
/var/log/ppp/0x12da2:  PC bitmap data, Windows 3.x format
/var/log/ppp/README.txt:  PNG image data,8-bit/color RGB
```

The search correctly identified three files, a *.jpg, a *.bmp, and a *.png, whose name and/or extension were changed in an effort to obfuscate their true type. This technique will work correctly as long as the files signature remains intact.

4. Conclusions and Future Work

The techniques described in this paper work well in identifying and recovering digital evidence for a large portion of the cases law enforcement agents will encounter. Changes in technology, particularly increases in storage capacity, are beginning to create problems for law enforcement agencies, however. For instance, the FBI computer analysis and response team (CART) saw a three-fold increase in cases from 1999 to 2003; the amount of data however increased 46-fold [3]. It is not uncommon for agents to encounter servers storing terabytes of data, equating to millions of documents, each of which is a potential piece of evidence. The critical question for law enforcement is: Which of the millions of digital artifacts is probative "evidence" and which is not? The techniques described in this chapter do not scale well to such tremendous data systems.

Although some forensic procedures are automated – such as the hash analysis and searches – many require manual input or human interpretation. In fact, almost no conventional digital forensic techniques scale well to terabyte-sized systems. As the amount of data grows, automated procedures for identifying, recovering, and examining digital evidence will be required to process evidence in a reasonable time period. Below we describe a taxonomy of digital artifacts that could serve as the basis for an automated system to identify probative evidence in large-scale systems. The taxonomy conceptualizes digital artifacts based on three attributes: (i) the artifact contents, (ii) its associated metadata, and (iii) ambient information. A digital artifact values for these attributes are both digital and identifiable, that is, knowing the identifier for a digital artifact (e.g., file name or inode number) one can identify, and therefore recover, the artifact contents, metadata, and ambient information. Consequently, it is conceivable that an automated procedure can be developed that is capable of recovering these values, obviating the need for any manual input or interpretations.

References

[1] B. Buckeye and K. Liston, Recovering deleted files in Linux (www.samag.com/documents/s=7033/sam0204g/sam0204g.htm), 2003.

[2] P. Craiger, Computer forensics procedures and methods, to appear in *Handbook of Information Security*, H. Bigdoli (Ed.), John Wiley, New York, 2005.

[3] P. Craiger, M. Pollitt and J. Swauger, Digital evidence and digital forensics, to appear in *Handbook of Information Security*, H. Bigdoli (Ed.), John Wiley, New York, 2005.

[4] A. Crane, Linux undelete how-to (www.praeclarus.demon.co.uk/tech/e2-undel/html/howto.html), 1999.

[5] S. Pate, *UNIX Filesystems: Evolution, Design and Implementation*, John Wiley, New York, 2003

[6] T. Warren, Exploring /proc (www.freeos.com/articles/2879/), 2003.

Chapter 20

DETECTING HIDDEN DATA IN EXT2/EXT3 FILE SYSTEMS

S. Piper, M. Davis, G. Manes and S. Shenoi

Abstract The use of digital forensic tools by law enforcement agencies has made it difficult for malicious individuals to hide potentially incriminating evidence. To combat this situation, the hacker community has developed anti-forensic tools that remove or hide electronic evidence for the specific purpose of undermining forensic investigations. This paper examines the latest techniques for hiding data in the popular Ext2 and Ext3 file systems. It also describes techniques for detecting hidden data in the reserved portions of these file systems.

Keywords: Anti-forensics, data hiding, file systems, Ext2/Ext3

1. Introduction

Digital forensics focuses on the identification, preservation, discovery and retrieval of electronic evidence [9]. The use of state-of-the-art forensic tools by law enforcement agencies has made it increasingly difficult for malicious individuals to hide potentially incriminating evidence. To combat this situation, the hacker community has experimented with anti-forensic techniques and tools [11]. Anti-forensics is defined as "the removal, or hiding, of evidence in an attempt to mitigate the effectiveness of a forensic investigation" [6]. Based on the number of anti-forensic tools available on the Internet [5], it is clear that anti-forensic techniques are being used by malicious individuals who wish to evade detection.

Recently, an individual known as *the grugq* unveiled the `Data Mule FS` software that conceals data within metadata structures of the Ext2 and Ext3 file systems [7]. `Data Mule FS` is specifically designed to hide data from forensic tools and file system checking software.

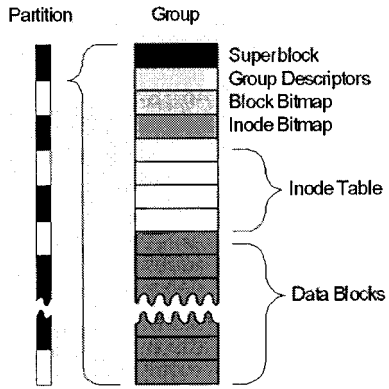


Figure 1. Ext2/Ext3 file system structure.

This paper examines the latest techniques for hiding data in the popular Ext2 and Ext3 file systems. It also describes techniques, including a utility named `rfinder`, which ensure that data hidden by anti-forensic programs such as `Data Mule FS` can be identified and retrieved by forensic investigators.

2. Ext2/Ext3 File Systems

A file system specifies how files, file metadata and other information are stored [4]. It allows the operating system to efficiently determine the locations of free space where new files can be stored. Also, it enables the operating system to quickly access and delete files.

Ext2 is the default file system [3, 15] for several Linux distributions [1]. It was created in 1993 to address limitations in earlier Linux file systems, such as Minix, which restricted file names to 14 characters and overall file system size to 64MB [3].

Ext2 has been upgraded to Ext3, which adds journaling, i.e., a means for recording hard drive accesses to minimize error-checking in the event of unexpected shutdowns [8, 13, 14]. Since Ext2 and Ext3 have the same structure, forensic techniques, anti-forensic techniques, and techniques for countering anti-forensic techniques are the same for both file systems.

Figure 1 shows the general layout of an Ext2/Ext3 file system, which comprises structures that are themselves segmented into smaller structures. The partitions are split into 8MB groups. Each group is divided into 1KB, 2KB or 4KB blocks, with the first few blocks containing metadata about the partition, group, and files within the group. The remaining blocks contain file data.

Block Bitmap (binary dump)

```
11111111 11111111 11111111 11111111 11111111 11100000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000001 11111111 11111111 11111111 11111111
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
...
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
```

Inode Table (hex dump)

```
Inode 0 (Bad Blocks Inode)
 00001400  00 00 00 00 00 00 00 00  6c 2c 17 40 6c 2c 17 40  |........l,.@l,.@|
 00001410  6c 2c 17 40 00 00 00 00  00 00 00 00 00 00 00 00  |l,.@............|
 00001420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 00001430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 00001440  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 00001450  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 00001460  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 00001470  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
Inode 1 (Root Inode)
 00001480  ed 41 00 00 00 04 00 00  09 2e 17 40 08 2e 17 40  |.A.........@...@|
 00001490  08 2e 17 40 00 00 00 00  00 00 03 00 02 00 00 00  |...@............|
 000014a0  00 00 00 00 00 00 00 00  1c 00 00 00 00 00 00 00  |................|
 000014b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 000014c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 000014e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 000014d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
 000014f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Figure 2. Block bitmap and inode blocks for a floppy disk.

The first block of the first group in a partition is called the "superblock." The superblock, which contains metadata about the partition, is essential to the operation of the file system, and its data is duplicated in superblocks in other groups for redundancy. Following the superblock are group descriptors that contain metadata about the current group and other groups in the partition (Figure 1). Specifically, group descriptors contain data about the number of free blocks in a group, and the locations of the block bitmap, inode bitmap and inode table.

Figure 2 presents a block bitmap for a floppy disk. The block bitmap indicates which blocks in the group are allocated, i.e., where file data are actually stored. Each block has a single bit in the bitmap, which is set

```
00007000  02 00 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |................|
00007010  0c 00 02 02 2e 2e 00 00  0b 00 00 00 14 00 0a 02  |................|
00007020  6c 6f 73 74 2b 66 6f 75  6e 64 00 00 0c 00 00 00  |lost+found......|
00007030  14 00 09 01 66 69 72 73  74 2e 74 78 74 00 00 00  |....first.txt...|
00007040  0d 00 00 00 14 00 0a 01  73 65 63 6f 6e 64 2e 74  |........second.t|
00007050  78 74 00 00 0e 00 00 00  ac 03 09 01 74 68 69 72  |xt..........thir|
00007060  64 2e 74 78 74 00 00 00  00 00 00 00 00 00 00 00  |d.txt...........|
00007070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Figure 3. Contents of a directory listing data block.

to one if the block is allocated and zero if it is not. Note that the bits at the end of the bitmap in Figure 2 are set to one. This is because these locations do not exist on a floppy disk; setting the corresponding bits to one (i.e., marking them as allocated) prevents the operating system from writing data to these blocks.

The inode is a smaller unit than a block that contains metadata about a particular file. Figure 2 shows a hex dump of the first two inodes of a file system. Each inode has a size of 128 bytes, and contains data about the file creation time, file size, and location of file data.

Data blocks make up the majority of the blocks in each group (typically 97%) and contain the data portions of files and directory listings [10]. Figure 3 shows the contents of a root directory. The directory listing stores the names of files/directories (`.`, `..`, `lost+found`, `first.txt`, `second.txt`, `third.txt`) in the directory. Also, the directory listing stores data about the inodes corresponding to the files, and the types of the files (i.e., file or directory).

3. Data Hiding Techniques

Designed for flexibility, file system components incorporate reserved locations to store additional metadata for future upgrades to the file system. Currently, applications do not read or write to reserved locations, and data written to these locations neither overwrites any useful data nor affects system operation. Therefore, the reserved locations are ideal sites for hiding data.

Figure 4 shows the source code for a group descriptor [2]. A total of 14 bytes in the group descriptor can be used to hide data: 2 bytes for the pad (`bg_pad`) and 12 bytes for the reserved variable (`bg_reserved[3]` is an array of three 4-byte words). This may not appear to be very much space; however, such reserved locations exist throughout the file system. A large file can be hidden (e.g., by `Data Mule FS`) by dividing it into smaller portions that are inserted in each of the reserved locations.

```
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 4. Source code for the group descriptor.

Reserved locations may be discerned by reviewing the kernel source code located in `./include/linux/ext2_fs.h` [2] or the source code for `e2fsprogs`, the project that maintains and updates the Ext2 file system [12].

Hiding data within a file system does not require the modification or addition of files. On the other hand, the use of encryption or steganography to hide data is detectable by MD5 hashes and other schemes for identifying file alterations. Therefore, data concealed within file system structures is likely to go undetected.

We experimented with the ability of digital forensic tools to find data concealed within file system structures. Data was written to various reserved locations, and the images of the file systems were analyzed using three popular forensic tools: EnCase v4.15, FTK v1.42 build 03.12.05, and iLook v7.0.35. The file system checking tool, `e2fsck`, which is standard on Linux, was used to test if the file systems behaved strangely or reported errors when data was hidden in their reserved locations. The tool was run with the `-f` argument to force checking. This bypassed any shortcuts that `e2fsck` might take when file systems are flagged as healthy.

In general, neither `e2fsck` nor any of the digital forensic tools produced specific alerts when data was hidden in reserved locations. These tools assume that the reserved locations are only used for file system data and that no other data are stored within their structures. However, as we show in Section 4, these tools can be used to locate hidden data, but one must know where to look.

Data may be hidden in several reserved locations within a file system. Procedures for hiding data in four of these reserved locations are described below.

Hiding Data in the First 1KB of a Partition

Ext2 does not use the first kilobyte of each partition because it assumes that this space might be used for a boot loader. This space is not used by the file systems on floppy disks and many hard-drive partitions.

Procedure:

1 Create a file named `secret_data.txt` with size less than 1KB.

2 Insert a floppy disk into the computer.

3 Execute the command: `# mke2fs -O none /dev/fd0` to format the disk as Ext2 with no extra features.

4 Execute the command: `# dd if=secret_data.txt of=/dev/fd0` to hide the data.

Alerts: None by `e2fsck`, EnCase, FTK and iLook.

Hiding Data in Reserved Inodes

Ext2 does not use inodes 7-10, opening up 512 bytes for hiding data. On an Ext2-formatted floppy disk, hidden data may start at byte 5888. The inode blocks start at `0x1400` and the first 6 inodes take up 6*128 = `0x300` bytes, so `0x1400` + `0x300` = `0x1700` = 5888. Note that Ext3 uses inodes 7 and 8, providing 256 bytes for hidden data starting at byte 6144.

Procedure:

1 Create a file named `secret_data.txt` with size less than 512 bytes.

2 Insert a floppy disk into the computer.

3 Execute the command: `# mke2fs -O none /dev/fd0` to format the disk as Ext2 with no extra features.

4 Execute the command: `# dd if=secret_data.txt of=/dev/fd0 seek=5888` to hide the data.

Alerts: "Bad mode" error reported by `e2fsck`. None by EnCase, FTK and iLook.

Hiding Data in Redundant Superblocks

Superblocks are repeated in groups on a disk. Depending on the specific Ext2/Ext3 implementation and distribution, there is a superblock in every group or a superblock in some groups (when the "sporadic superblock" flag is set). It is possible to hide 1K to 4K of data in each redundant superblock. Note that a floppy disk has only one superblock (i.e., it has no redundant superblocks); therefore, data cannot be hidden in a floppy disk using this procedure.

Procedure:

1 Create a file named `secret_data.txt` with size between 1K to 4K (depending on the file system block size).

2 Execute the command: `# mke2fs -O none /dev/`*hd* to format a partition of hard drive *hd* (e.g., `hda1` or `hdb1`). The option `-O none` ensures that the sporadic superblock flag is not set.

3 Locate a redundant superblock by comparing blocks to a known superblock, such as the first superblock located at offset `0x400`. The redundant superblocks will also have signature values of `0x53ef` offset by `0x38` bytes from the beginning of the block.

4 Execute the command: `# dd if=secret_data.txt of=/dev/`*hd* `seek=`*loc* to hide the data. Note that *loc* is the starting byte of the redundant superblock.

Alerts: None by `e2fsck`, EnCase, FTK and iLook.

Hiding Data in Reserved Portions of Superblocks
Every superblock has at least 384 bytes reserved for updates to the file system specification. Since this group of bytes does not currently have a purpose, it can be used to hide data. Also, a superblock is supposed to fit in only 1K of space, but it uses an entire block on the file system (up to 4K). Therefore, it is often possible to hide data in the extra 3K (max) of space.

Procedure:

1 Create a file named `secret_data.txt` with size less than 384 bytes.

2 Insert a floppy disk into the computer.

3 Execute the command: `# mke2fs -O none /dev/fd0` to format the disk as Ext2 with no extra features.

4 Execute the command: `# dd if=secret_data.txt of=/dev/fd0 seek=1664` to hide the data.

Alerts: None by `e2fsck`, EnCase, FTK and iLook.

The versions of EnCase, FTK and iLook used in the experiments had the means to access and view the hidden data. However, none of the tools produced alerts that data was hidden in the reserved portions of the file systems. Forensic tools are designed to prevent investigators from having to examine data manually with a hex editor. As Figure 5 demonstrates for the EnCase tool, an investigator must use the built-in hex editor of the tool to discover the hidden data.

4. Data Detection Techniques

This section describes how data hidden in the reserved portions of the Ext2/Ext3 file systems may be detected. First, the application of digital forensic tools is discussed. Next, the use of our data detection utility (`rfinder`) is explained. Finally, the use of a file system checker is discussed.

4.1 Digital Forensic Tools

The digital forensic tools considered in this work (EnCase v4.15, FTK v1.42 build 03.12.05, and iLook v7.0.35) are not designed to discover hidden data in the reserved portions of the Ext2/Ext3 file systems. How-

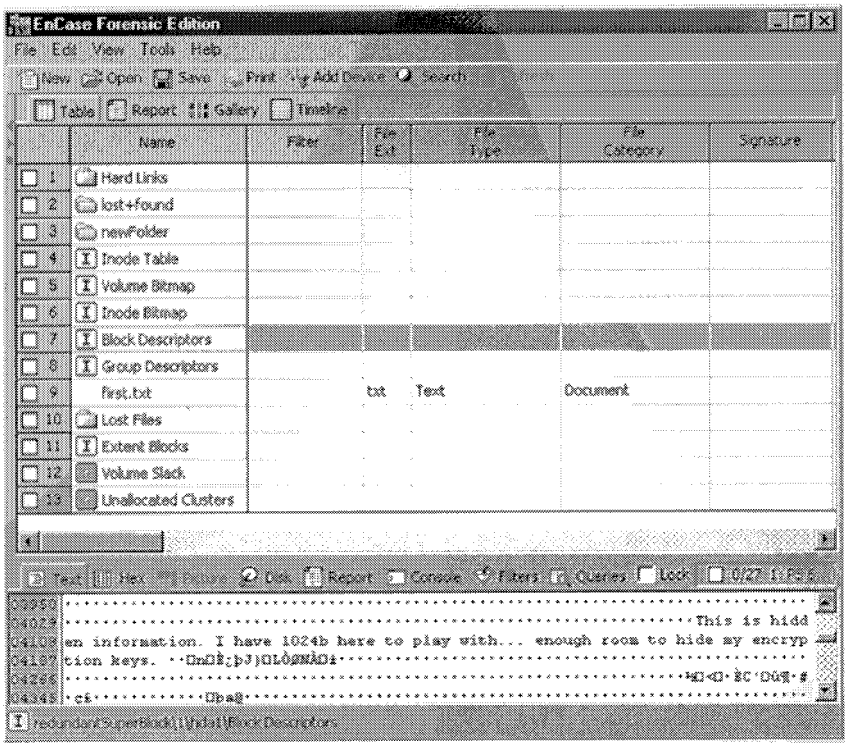


Figure 5. EnCase view of data in a redundant superblock showing hidden data.

ever, they can be used to identify hidden data if one knows where to look.

Detecting Hidden Data in the First 1KB of a Partition

Hidden data in the first 1KB of a partition can be found by looking under "Block Descriptors" for EnCase, "Boot Record" for FTK, and "Disk View" for iLook. The presence of non-zero values potentially indicates the existence of hidden data.

Detecting Hidden Data in Reserved Inodes

Hidden data in reserved inodes can be found by looking under "Inode Table" for EnCase, "Inode Table" for FTK, and "Disk View" for iLook.

Detecting Hidden Data in Redundant Superblocks

Hidden data in redundant superblocks can be found by looking under "Block Descriptors" for EnCase (see Figure 5), "Superblocks" for FTK, and "Disk View" for iLook.

Detecting Hidden Data in Reserved Portions of Superblocks
Hidden data in the reserved portions of superblocks can be found by looking under "Block Descriptors" for EnCase, "Superblocks" for FTK, and "Disk View" for iLook.

4.2 Data Detection Utility

It is possible to capitalize on the normal operation of a file system to discover hidden data. When a partition is formatted as Ext2 or Ext3, many of the reserved locations are zero wiped. This means that non-zero values that appear in the reserved locations are likely to be hidden data. We have created the `rfinder` utility that automatically searches through the reserved locations of Ext2/Ext3 file systems for non-zero values. If any non-zero values are found, `rfinder` displays a hex dump of the corresponding data.

Figure 6 shows the output obtained by running the `rfinder` utility on a 1GB partition. The partition has no hidden data in it, but if it did, a hex dump of the data would have been displayed. The hex dump of the first redundant superblock is displayed by `rfinder` because this superblock is different from the original superblock. This is common because only the first superblock is updated, for example, when the partition was last mounted. The last line of `rfinder`'s output advises the user to run the file system checker, `fsck`, on the partition to continue the search for hidden data.

4.3 File System Checkers

When the `fsck` file system checker is run on an Ext2/Ext3 partition, a utility called `e2fsck` is invoked to specifically check the Ext2/Ext3 file system. This utility can help discover whether or not other methods have been used to hide data.

The command: `e2fsck -f <image_file>` forces `e2fsck` to perform all of its checking. The image is modified slightly when `e2fsck` attempts to repair the file system. Therefore, a copy of the image must be made before executing the command. The modifications made by `e2fsck` can then be resolved using the `diff` command to compare the original and modified images. Note that `e2fsck` will flag everything that may be wrong with the file system, and not just hidden data.

Although `e2fsck` was originally created to locate faults in file systems, it is very effective for finding hidden data. For example, if certain blocks are marked as being in use even when no file uses them, `e2fsck` will identify and remedy this situation by de-allocating the blocks. This does not necessarily imply that data is hidden there. However, allocating and

```
--rfinder v0.92.2--  Author: Scott Piper

Checking /dev/hdb1...
Found an Ext2FS disk with a 4KB block size and 12 groups on the disk.

Checking the first 1K...
Nothing found (all zeroes)

Checking the superblock s_padding1 var...
Nothing found (all zeroes)

Checking the reserved portions of the first superblock (sometimes contains
        data, and will raise a false positive)...
Nothing found (all zeroes)

Checking the reserved inodes...
Nothing found (all zeroes)

Checking reserved portions of the inodes...
Nothing found (all zeroes)

Checking redundant superblock in group: 1
**This block is different the previous** (this is the first redundant super
        block though, so this is normal)
0x8000000: 20 07 03 00 60 0D 06 00 78 4D 00 00 FC F4 05 00  | ...`...xM......|
0x8000010: 15 07 03 00 00 00 00 00 02 00 00 00 02 00 00 00  |................|
0x8000020: 00 80 00 00 00 80 00 00 A0 3B 00 00 00 00 00 00  |.........;......|
0x8000030: 9D 1C F0 41 00 00 27 00 53 EF 00 00 01 00 00 00  |...A..'.S.......|
0x8000040: 9D 1C F0 41 00 4E ED 00 00 00 00 00 01 00 00 00  |...A.N..........|
0x8000050: 00 00 00 00 0B 00 00 00 80 00 01 00 00 00 00 00  |................|
0x8000060: 02 00 00 00 01 00 00 00 96 9E 7C 2F D9 EA 4D A9  |..........|/..M.|
0x8000070: BF BA 0F 8D 9A 35 D6 D1 00 00 00 00 00 00 00 00  |.....5..........|
0x8000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
0x8000090: *
0x80000E0: 00 00 00 00 00 00 00 00 00 00 00 00 AB 8B C3 27  |...............'|
0x80000F0: C5 A0 44 42 85 2E 9B 9F DC 61 C6 13 02 00 00 00  |..DB.....a......|
0x8000100: 00 00 00 00 00 00 00 00 9D 1C F0 41 00 00 00 00  |...........A....|
0x8000110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
0x8000120: *
0x8000FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00] |................|

Checking reserved group descriptor space in group: 1
Checking redundant superblock in group: 3       (same as previous)
Checking reserved group descriptor space in group: 3
Checking redundant superblock in group: 5       (same as previous)
Checking reserved group descriptor space in group: 5
Checking redundant superblock in group: 7       (same as previous)
Checking reserved group descriptor space in group: 7
Checking redundant superblock in group: 9       (same as previous)
Checking reserved group descriptor space in group: 9

Program completed successfully.
To continue looking for errors, run "fsck -f /dev/hdb1"
```

Figure 6. Output of the **rfinder** utility for a 1GB partition.

writing data to unused blocks is a technique that may be used to hide data, and `e2fsck` can help identify this situation.

5. Conclusions

Data hiding techniques have advanced from hiding secret files in folders that begin with a period (e.g., `./.hidden`) to secreting file fragments in obscure locations within file systems. Unlike other techniques, e.g., cryptography and steganography, hiding data within a file system does not involve the modification or creation of files, and is, therefore, not detectible by comparing hash values. Meanwhile, hacker tools like `Data Mule FS` are being used by malicious individuals to hide data from forensic tools and file system checking software.

The data hiding methods described in this paper are intended to expose the digital forensics community to anti-forensic techniques. Several methods for detecting hidden data are proposed, along with the `rfinder` utility, for countering anti-forensic tools such as `Data Mule FS`. While the strategies for combating anti-forensic techniques are discussed in the context of the Ext2 and Ext3 file systems, the underlying ideas are applicable to other file systems, including FAT and NTFS.

References

[1] D. Bovet and M. Cesati, *Understanding the Linux Kernel*, O'Reilly, Sebastopol, California, 2002.

[2] R. Card, Cross-referencing Linux (lxr.linux.no/source/include /linux/ext2_fs.h?v=2.6.10).

[3] R. Card, T. Ts'o and S. Tweedie, Design and implementation of the Second Extended File System, *Proceedings of the First Dutch International Symposium on Linux*, 1994.

[4] B. Carrier, *File System Forensic Analysis*, Addison-Wesley, Crawfordsville, Indiana, 2005.

[5] A. Cuff, Anti-forensic tools, Computer Network Defence Ltd., Corsham, Wiltshire, United Kingdom (www.networkintrusion.co.uk/foranti.htm), 2004.

[6] The grugq, Defeating forensic analysis on Unix, *Phrack 59* (www.phrack.org/show.php?p=59&a=6), July 28, 2002.

[7] The grugq, The art of defiling, presented at the *2004 Hack in the Box Conference* (packetstormsecurity.nl/hitb04/hitb04-grugq.pdf), October 8, 2004.

[8] M. Johnson, Red Hat's new journaling file system: Ext3 (www.redhat.com/support/wpapers/redhat/ext3/index.html), 2001.

[9] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*, Addison-Wesley, Boston, Massachusetts, 2002.

[10] D. Phillips, A directory index for Ext2, *Proceedings of the Fifth Annual Linux Showcase and Conference*, 2001.

[11] A. Saita, Antiforensics: The looming arms race, *Information Security Magazine*, May 2003.

[12] T. Ts'o, E2fsprogs: Ext2 file system utilities (e2fsprogs.sourceforge.net).

[13] S. Tweedie, Journaling the Linux Ext2fs filesystem, presented at the *Fourth Annual Linux Expo* (jamesthornton.com/hotlist/linux-filesystems/ext3-journal-design.pdf), 1998.

[14] S. Tweedie, Ext3: Journaling filesystem (olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html), July 20, 2000.

[15] M. Wilcox, The Second Extended File System (mail.nl.linux.org/kernel-doc/1999-03/msg00001.html), March 1, 1999.

VI

APPLICATIONS AND TECHNIQUES

Chapter 21

FORENSIC ANALYSIS OF DIGITAL IMAGE TAMPERING

Gilbert Peterson

Abstract The use of digital photography has increased over the past few years, a trend which opens the door for new and creative ways to forge images. The manipulation of images through forgery influences the perception an observer has of the depicted scene, potentially resulting in ill consequences if created with malicious intentions. This poses a need to verify the authenticity of images originating from unknown sources in absence of any prior digital watermarking or authentication technique. This research explores the ability to detect image forgeries created using multiple image sources and specialized methods tailored to the popular JPEG image format. Four methods are presented for detection of image tampering based on fundamental image attributes common to any forgery. These include discrepancies in (i) lighting levels, (ii) brightness levels, (iii) underlying edge inconsistencies, and (iv) anomalies in JPEG compression blocks. These methods detected image forgeries with an observed accuracy of 60% in a completely blind experiment containing a mixture of 15 authentic and forged images.

Keywords: Image forgery, image forensics, image authentication

1. Introduction

Digital technologies allow for manipulation in photographic development; thereby making it necessary to verify the authenticity of a digital image. As digital cameras become more prevalent and accepted at an evidentiary level, an individual's conviction may depend on the authenticity of a digitalimage. The traditional technique for declaring image propriety and subsequently authentication applies a visible or invisible watermark [3] immediately after capture. Checking the presence of the watermark on the image verifies its authenticity. This procedure requires the image originate from a known and authenticating source.

This paper presents four techniques for detecting tampering in JPEG compressed images given images from unknown sources. These techniques consider the color and brightness of individual pixels as well as the JPEG image format. These techniques are then applied in a blind test on a set of 15 images consisting of real and expert forged images.

2. Related Work

This section discusses the JPEG digital image format and existing research in image forgery detection. To assist in this discussion forged image detection is separated into two classes, copy-move and copy-create. The reason for distinguishing classes of image forgeries is because some image processing techniques are better suited to a specific class.

2.1 JPEG Image Format

Digital image compression and storage fall into two categories, lossless and lossy. In lossless compression, techniques like GIF, TIFF and PNG, the image quality is maintained resulting in the uncompressed image being identical to the pre-compressed image. For lossy compression techniques like JPEG, the quality of the image is sacrificed for a smaller storage size.

Lossy JPEG compression exploits the fact that the human eye is less sensitive to higher frequency information (e.g., edges and noise) in an image than to lower frequencies. The jpeg encoding process [13], Figure 1, starts by breaking the raw image into blocks, usually sized to 8×8 pixels. A total of 64 Discrete Cosine Transform (DCT) coefficients are computed for each block, converting the block from the spatial domain to the frequency domain. The higher frequency DCT coefficients are then rounded off according to the values of the quantization matrix, which determines the tradeoff balance between image quality and compression ratio, also termed the quality factor. The matrix of quantized DCT coefficients is then encoded into a binary stream with lossless Huffman compression. An image is extracted from a jpeg file by reversing this process.

2.2 Copy-Move Forgery Detection

The first class of image forgeries includes images tampered by means of copying one area within an image and pasting it onto another, copy-move forgeries. Figure 2 illustrates an example in which copied parts of the foliage cover and mask the truck to completely hide it.

Existing methods developed to detect this type of forgery build on the intuitive suggestion of performing an exhaustive comparison search.

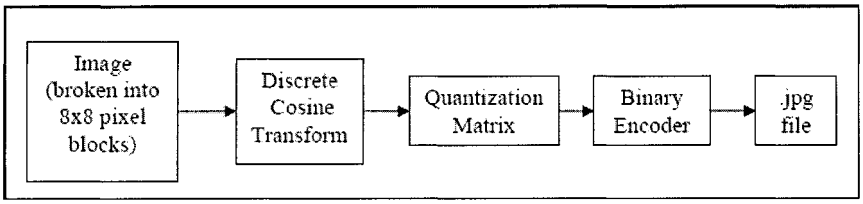


Figure 1. JPEG compression process.

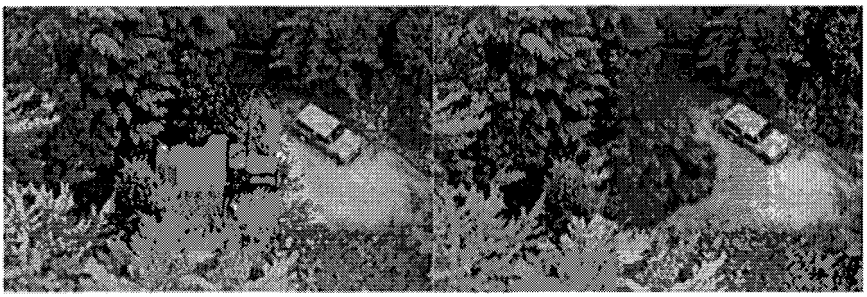


Figure 2. Example of copy-move image forgery [6].

Fridrich, *et al.* [6] overlay each circularly shifted position of the grayscale converted image, comparing it with the original to yield the areas copied and pasted. An improvement on the computational complexity is a block matching variation using a B×B block of pixels, which represents the minimal size considered for a match. This technique reduces the computational complexity of the technique and also dictates the desired accuracy of the image in question.

The application of block matching to lossy JPEG images makes use of blocks matched based on their representation consisting of quantized DCT coefficients. In this method, the same technique is used which creates a matrix from B×B blocks. The difference being the storage of computed DCT coefficients instead of pixel values [6].

2.3 Copy-Create Forgery Detection

The second class of forged images deals with creating the forgery by taking one or more images and copying and pasting from various areas within each to form a forged image. The image processing community refers to this as an image "composition," which is defined as the "digitally manipulated combination of at least two source images to produce an integrated result" [2]. The name for these types of images, in context

of this article, is copy-create forgeries. Figure 3 shows how the three images at the bottom can be merged into a single image.

Two methods currently exist for detecting copy-create forgeries, edge detection algorithms and spectral analysis. Edge detection techniques attempt to detect double or "ghost" edges around objects in the environment caused by the blurring of space around the tampered objects [8]. Alternatively, spectral analysis approaches utilize Discrete Fourier Transforms (DFTs) and their ability to detect brightness and intensity levels of an image to detect variations caused by resampling [5, 8].

Figure 3. Example of image forgery created from several sources [6].

An edge is an area in the image where the intensity of pixels moves from a low value to a high value or vice versa [9]. Edge detection in images is conducted by convolving first-order operators with the image in order to locate areas that are discontinuous. Previous masks used in analyzing images were the Roberts, Sobel and Prewitt masks [8].

Forged images that are the result of merging two or more host images together usually requires that at least one image be cropped, resized, or rescaled. This manipulation leads to underlying changes in the statistical nature of the image, which spectral analysis captures. By calculating the discrete Fourier transform (DFT) of suspected areas of manipulation in the image, the analyst looks for a periodic pattern and local maximums suggesting that an area has been re-sampled [8].

Farid and Popescu [5] extend the spectral analysis approach by calculating a high-pass filtered "probability map" of the forgery, and then filtering the image to gain high detection accuracy. The probability map is calculated as a correlation between pixel neighbors estimated against several periodic samples, thereby removing the low frequency noise from the image which may return false positives. In the forgery detection algorithm, areas of this probability map are blocked off and used for comparison. One blocked area should encompass the suspected tampered portion and a second blocked area should cover an assumed authentic region [5].

Spectral analysis has been shown to work best on uncompressed or losslessly compressed images and requires the analyst to already anticipate where in the image the forgery exists. Images saved in the lossy JPEG format with quality factors less than 97 exhibit much lower detection accuracy, becoming a hit or miss occurrence [5]. It should be noted that most JPEG images are generally set to a quality factor of approximately 80/100 for optimal high quality, with medium to low quality images using much lower quality factors.

3. Analyzing JPEG Images

A person's expectation of an image is sometimes the best detection method in determining if an image is forged. As, the human eye usually picks up on copy-create forgeries because this type of forgery consists of several images, each of which may have different lighting, color patterns, quality, or shadows.

The first two techniques attempt to assist the analyst's eye by augmenting these differences, targeting the luminance and HSV values of the images. The third technique builds on the ideas behind convolution masks augmenting the double edge present in copy-create forgeries. The final technique examines the compression of the different JPEG compression blocks, searching for variations on the assumption that in a copy-create image the source images may have different quality factors.

3.1 Luminance Levels

The luminance of an image is the measurement of the perceived brightness levels [11]. Intuitively, if two images are taken from different cameras with different lighting, some sort of discrepancy may occur in those areas which were copied and pasted. In particular, analyzing a forged image looks for areas that are approximately the same distance away from the lens but have different luminance levels. This analysis is heavily dependant on the skill level of the person creating the forgery and

the resources available to perform the manipulation. Newer versions of image processing software make it easy for even a novice user to create forgeries based on automated "auto-brightness" adjustments.

The luminance level detector converts a color image to grayscale and then to binary by setting pixels 'on' if they exceed a user set luminance threshold and 'off' otherwise. The luminance threshold is a value between 0.0 and 1.0. To determine an appropriate threshold a value of approximately 0.50 is a good starting point with subsequent tests performed in both directions. One could also choose to use Otsu's method for finding greyscale thresholding values which minimizes the intraclass variance between black and white pixels [10]. The ultimate goal is to look for results depicting an area of suspected tampering, which are witnessed by unnatural or abnormal luminance levels in an area. Figure 5 shows the luminance results of Figure 4 based on a luminance threshold of 0.60, and revealing an abnormal pattern in the tampered area.

Figure 4. Tampered Lena Image.

3.2 Hue-Saturation-Value (HSV)

The hue of a color is described as the "tint," saturation or "shade" is the level of purity or intensity of a color; the value is the level of brightness or how light or dark it is [11]. As with luminance, if an area of an image is copied and pasted from a different source, the color and brightness, as captured from each respective image, may be different.

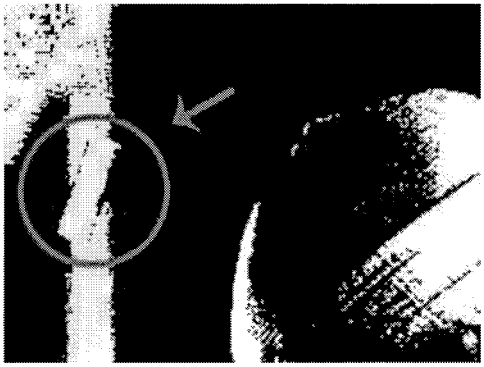


Figure 5. Result of luminance level test on forged Lena image.

Thorough analysis of a color image converted to HSV levels [12] helps determine this.

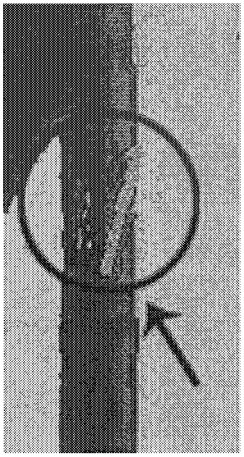


Figure 6. Result of converting forged Lena image into HSV color-space.

Figure 6 shows the results of a HSV color-space test performed on Figure 4. Again, the magnified area in this figure illustrates the tampered portion by showing an uneven color pattern and shape compared with the surrounding area. The abnormal color "bleeding" also indicates some form of tampering has occurred.

3.3 Alternative Filtering Mask

Several convolution filtering methods were analyzed by Lukas [8], including the Roberts, Sobel, Prewitt and Marr masks. These methods

have been limited in their detection of image forgeries due to their targeting of specific types of edges. Since what is of interest in forgery detection is not in detecting edges but in image discrepancies such as double edges, a custom convolution mask is created which places emphasis on a particular image's distinct contrasts. The created mask uses a 3×3 block size which is the best size for capturing the trends in an image without introducing too much pixel variation.

$$\begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}$$

The weight of 12 is placed on the center pixel along with all other neighbors' weights summing to -12. This filters out all areas in an image that are similar and magnifies those that vary greatly. These varying areas arise from prominent edges, and locations victim to image tampering. The analyst then looks for portions within the image that are noisy or contain "hidden" and "ghost" edges. Figure 7 shows this filtering method on Figure 4. In this example, the magnified portion shows the tampered area which exhibits a distinctive abnormal pattern in comparison with the surrounding area.

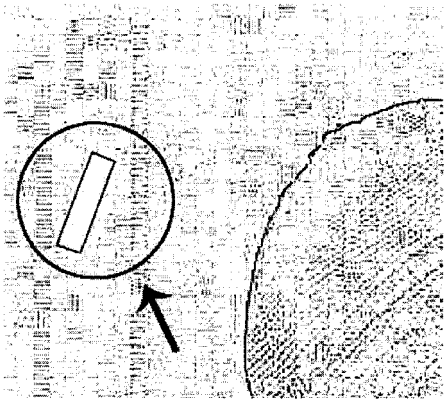


Figure 7. Inverted result of performing custom filter mask on forged Lena image.

3.4 JPEG Compression Forgery Detection

During the JPEG compression process (Figure 1), the image is broken into disjoint 8×8 blocks. These blocks then form a "fingerprint" of the image. When creating a copy-create forgery, it is composed of several pieces of other images which are cropped, scaled, and rotated to make the forged image's authenticity more believable. These pieces may have

originated from images that have previously been JPEG compressed with differing quality factors (QF).

This technique analyzes a JPEG image with respect to the 8×8 blocks used by the JPEG compression scheme and detects these QF differences. Performing a calculation on the boundaries of these blocks builds upon the technique presented by Fan and Queiroz [4] for detecting prior JPEG compression in a BMP image. Figure 8 shows an abstract representation of an 8×8 block of pixels in a JPEG image with letters representing interested pixel values.

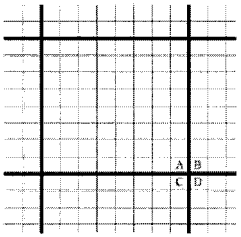


Figure 8. Abstract representation of an 8×8 block used by JPEG compression.

The calculation of $R(i,j) = |A - B - C + D|$ for each 8×8 block intersection, Figure 8, represents the degree of pixel variation present between the 8×8 block and its 3 neighbors. Variations in the block differences between image area are the result of differences in the compression levels across the image. To verify a suspected image of forgery, all $R(i,j)$ values are calculated for each block. Each block is then white if $(|R(i,j) - R(i,j+1)| > t) \vee (|R(i,j) - R(i+1,j)| > t)$ where t is a user definable threshold. This compares the intersection difference between the intersection to the right and to the bottom with black blocks indicating a large variation in the compression levels between intersections.

Figure 9 illustrates the proposed JPEG Block Technique using a threshold of 15. The result of the block analysis technique has uncovered a definitive pattern in the differing compression levels of the image. This is a good example of how the naked eye is fooled by the authenticity of a forged image, but the "fingerprint" of the JPEG compression scheme leaves pixel level differences.

The determination of the proper threshold starts with a value equal to 50. The result should then be analyzed with further testing using threshold values in increments/decrements of 5 or 10. Each test should look for distinctive patterns in the binary image or focus on areas suspected of tampering. As the threshold value decreases, the black pixels center on areas of image tampering. This is because high levels of JPEG block variability are usually seen in areas with prominent edges or that

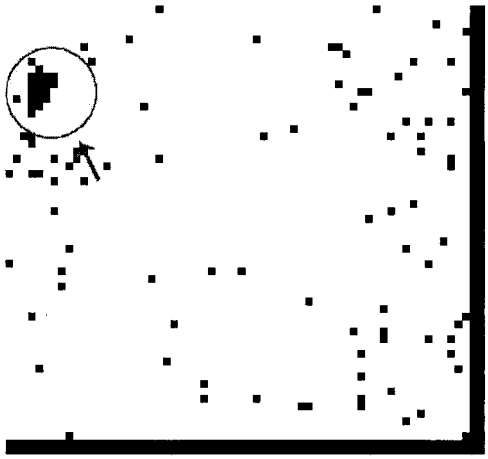


Figure 9. Result of performing JPEG block test on forged Lena image.

have been digitally tampered. The alternative occurs when the threshold is raised, the white pixels center on the tampered area which was pasted from a higher quality factor image.

4. Results

In order to obtain objectivity in testing the methods, the techniques are tested on a set of 15 images consisting of real and expert forged images where no information is provided about the authenticity of the images. For this test, each of the methods is applied to an image, for the luminance and JPEG compression forgery detection methods, the thresholds are adjusted in the effort of verifying a forged area. An image is declared a forgery if one of the techniques definitively demonstrates that there is an anomaly present.

Overall, 6 of the 15 test images were found to be incorrectly identified. This included 2 identified as false positive and 4 as false negatives. Therefore, an overall observed accuracy of this experiment is 60% with a 13.33% false positive result and 26.67% false negative result. It is interesting to note that the two images that were false positives were both trick camera shots, one failed the luminance and HSV tests was a night photograph with a very slow shutter speed. The other failed the JPEG compression detection was a photograph taken with a fisheye lens.

The results of this experiment raise some important points about performing the proposed methods to detect image tampering. When performing each technique on an image of unknown origin, some subjective analysis is required of each method's result. In the case of JPEG images

with low quality factors, one has to determine if a flagged area is due to actual image tampering or if high compression introduced the distortion, as can be the case with many images found on the web. Also, it is preferable to get a second opinion of each result to aid in the decision making process. This experiment overall proved to be interesting and found a respectable accuracy percentage compared to declaring authenticity without the help of any detection methods.

5. Conclusions

The detection of image tampering relies on one assumption, that the tampering performed by a forger introduces some detectable anomaly. This can be some inconsistent color or brightness pattern, abnormal edge, or other by-product of image tampering.

The four techniques presented in this paper extend image authentication to provide verification methods for the previously uninvestigated area of copy-create image forgeries in the lossy JPEG compression format. The JPEG compression detection method makes use of the JPEG "fingerprint" to determine if an image is a forgery. Subsequently, the other three methods developed work on any digital image due to their specialization in fundamental attributes of any digital image.

Testing these four methods in a blind experiment of 15 authentic and expert forged JPEG images revealed a detection accuracy of 60%. Detection accuracy was found to be heavily dependent on the amount of time spent analyzing the results of each method as well as any pre-existing tampering knowledge of the image in question.

During the testing and development for this research no one technique was found to be best at detecting every image forgery and enforces the idea that a multilayered approach is required for image authentication. Additionally, the ability to detect a forgery is tied to the amount of creativity and effort of the forger given there are an infinite number of possibilities to create, alter, and digitally manipulate any given image. Some of the methods a forger could employ to avoid detection are to manipulate the luminance and HSV levels to match the remainder of the image, and perform the manipulation on a larger lossless image that is then compressed on completion.

6. Acknowledgements

This work paper was supported by the Digital Data Embedding Technologies group of the Air Force Research Laboratory, Information Directorate. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright no-

tation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Air Force Research Laboratory, or the U.S. Government.

References

[1] Associated Press, Britain says soldier held in photo probe, *Newsday*, May 18, 2004.

[2] R. Brinkmann, *The Art and Science of Digital Compositing*, Academic Press, San Diego, California, 1999.

[3] R. Chandramouli, R. Memon and M. Rabbani, Digital watermarking, in *Encyclopedia of Imaging Science and Technology*, J. Hornak (Ed.), John Wiley, New York, 2001.

[4] Z. Fan and R.L. de Queiroz, Identification of bitmap compression history: JPEG detection and quantizer estimation, *IEEE Transactions on Image Processing*, vol. 12(2), pp. 230-235, 2003.

[5] H. Farid and A. Popescu, Exposing digital forgeries by detecting traces of resampling, *Proceedings of the IEEE Transactions on Signal Processing*, 2004.

[6] J. Fridrich, J. Lucas and D. Soukal, Detection of copy-move forgery in digital images, *Proceedings of the Digital Forensics Research Workshop*, 2003.

[7] K. Guggenheim, New prison abuse photos outrage lawmakers, *Phillyburbs*, May 13, 2004.

[8] J. Lukas, Digital image authentication using image filtering techniques, *Proceedings of the Fifteenth Conference of Scientific Computing*, 2000.

[9] C.M. Luong, *Introduction to Computer Vision and Image Processing*, Department of Pattern Recognition and Knowledge Engineering, Institute of Information Technology, Hanoi, Vietnam, 2004.

[10] N. Otus, A threshold selection method from gray-level histograms, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9(1), pp. 62-66, 1979.

[11] J. Sachs, *Digital Image Basics*, Digital Light & Color, Cambridge, Massachusetts, 1999.

[12] A. Smith and E. Lyons, HWB – A more intuitive hue-based color model, *Journal of Graphics Tools*, vol. 1(1), pp. 3-17, 1996.

[13] Society for Imaging Science and Technology, Jpeg tutorial (www.imaging.org/resources/jpegtutorial/index.cfm).

Chapter 22

CONTENT-BASED IMAGE RETRIEVAL FOR DIGITAL FORENSICS

Y. Chen, V. Roussev, G. Richard III and Y. Gao

Abstract Digital forensic investigators are often faced with the task of manually examining a large number of (photographic) images to identify potential evidence. The task can be daunting and time-consuming if the target of the investigation is very broad, such as a web hosting service. Current forensic tools are woefully inadequate: they are largely confined to generating pages of thumbnail images and identifying known files through cryptographic hashes. This paper presents a new approach that significantly automates the examination process by relying on image analysis techniques. The strategy is to use previously-identified content (e.g., contraband images) and to perform feature extraction, which captures mathematically the essential properties of the images. Based on this analysis, a feature set database is constructed to facilitate automatic scanning of a target machine for images similar to the ones in the database. An important property of the approach is that it is not possible to recover the original image from the feature set. Therefore, it is possible to build a (potentially very large) database targeting known contraband images that investigators may be barred from collecting directly. The approach can be used to automatically search for case-specific images, contraband or otherwise, and to provide online monitoring of shared storage for early detection of specific images.

Keywords: Digital forensics, image analysis, image retrieval

1. Introduction

Digital forensic investigations often require the examination of pictures found on target media. Two typical tasks are the identification of contraband images and the identification of case-specific images, the presence of which can establish a fact or a logical link relevant to the investigation. The essential problem is that current forensic tools are often ill-equipped to deal with the scale of the task. To illustrate, we

recently recovered approximately 34,000 image files on a randomly selected machine in our general-purpose computing laboratory. Note that this was a relatively old system with a very modest 6 GB hard drive and the images were mostly stored in the browser's cache. Even if an investigator were to spend a fraction of a second on each image, it would still require several hours to browse through all the images. The dramatic drop in prices of storage devices coupled with the leap in capacity (a 200 GB hard drive now costs about $100), will make the examiner's task even more difficult by removing any incentive for users to delete images. Thus, it is not unreasonable to expect that the hard drive of a home user could contain hundreds of thousands of images, while a web hosting service can have tens of millions of images. Clearly, examining all these images is virtually intractable, and investigators will need some means to narrow the search space.

The driving problem behind this work has been the identification of contraband images. This task consumes a significant fraction of the resources of our partners at the Gulf Coast Computer Forensics Laboratory (GCCFL). They have a clear and pressing need for a forensic tool that would allow the automated examination of images on a massive scale. Similar problems in traditional forensics (e.g., fingerprint identification) have been tackled by building large reference databases that allow evidence from previous cases to be automatically searched. Clearly, a system capable of automatically identifying contraband images on target media by cross referencing a database of known images could be of significant help to investigators. The problem, however, is that unlike other forensic artifacts, contraband images typically cannot be stored, even by law enforcement agencies, for future reference. Aside from the legal barriers, building a sizeable reference database to be used routinely by numerous agencies would be a challenging task. The storage and bandwidth requirements would be staggering. Scalability would be difficult to achieve as the replication and distribution of such highly sensitive images would have to be limited. Finally, a security breach at an image storage facility or misuse by authorized personnel could have serious implications.

A well-designed system should not rely on having access to the original images during the lookup process. Rather, it should have a single opportunity to access the original when it can extract and store some identifying ("fingerprint") information for later reference. Clearly, the fingerprint information should permit a high-probability match; but it should also be impossible to reconstitute any recognizable version of the original image. We believe that analytical methods for content-based image retrieval can address image analysis needs in digital forensics. This

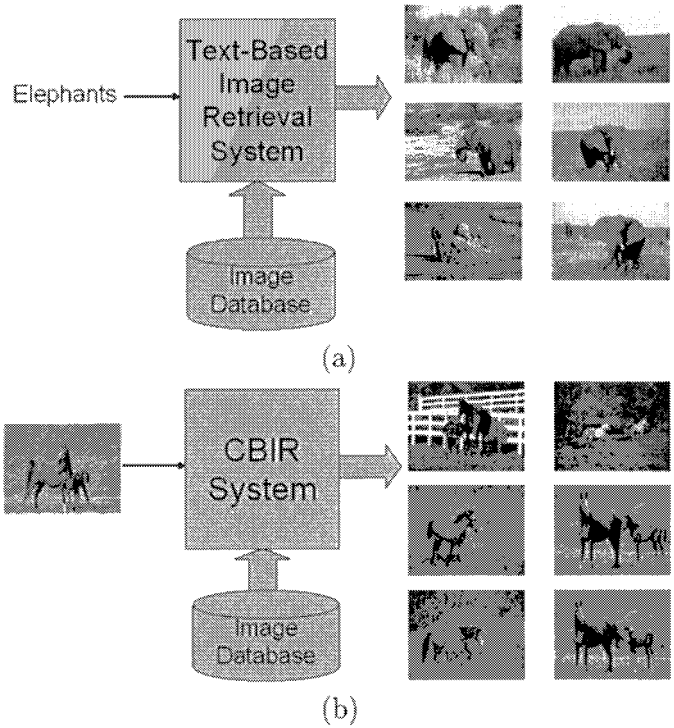


Figure 1. Schematic diagrams of: (a) text-based image retrieval system; (b) content-based image retrieval system.

paper is a first effort to evaluate the suitability of the approach and to present an architectural framework that would allow the deployment of a working system.

The following section describes previous work in content-based image retrieval, which forms the basis of our own work. Next, a set of experimental results is presented to validate the use of content-based image retrieval in digital forensic investigations. Finally, an architectural design (currently under implementation) is specified.

2. Content-Based Image Retrieval

2.1 Overview

Depending on their query formats, image retrieval algorithms roughly fall into two categories: text-based approaches and content-based methods (see Figure 1). Text-based approaches associate keywords with each stored image. These keywords are typically generated manually. Image retrieval then becomes a standard database management problem. Some

commercial image search engines, e.g., Google Image Search and Lycos Multimedia Search, are text-based image retrieval systems. However, manual annotation for a large collection of images is not always available. Furthermore, it may be difficult to describe image content with a small set of keywords. These issues have motivated research on content-based image retrieval (CBIR), where retrieval of images is guided by providing a query image or a sketch generated by a user (e.g., a sketch of a horse).

Many CBIR systems have been developed over the past decade. Examples include the IBM QBIC System [4], the MIT Photobook System [12], the Berkeley Chabot [11] and Blobworld Systems [1], the Virage System [7], Columbia's VisualSEEK and WebSEEK Systems [14], the PicHunter System [2], UCSB's NeTra System [9], UIUC's MARS System [10], the PicToSeek System [6] and Stanford's WBIIS [16] and SIMPLIcity systems [15].

From a computational perspective, a typical CBIR system views the query image and the images in the database as a collection of features, and ranks the relevance between the query and any matching image in proportion to a similarity measure calculated from the features. The features are typically extracted from shape, texture, intensity or color properties of the query image and the images in the database. These features are image signatures and characterize the content of images, with the similarity measure quantifying the resemblance in content features between a pair of images.

Similarity comparison is an important issue. In general, the comparison is performed either globally, using techniques such as histogram matching and color layout indexing, or locally, based on decomposed regions (objects). As a relatively mature method, histogram matching has been applied in many general-purpose image retrieval systems, e.g., IBM QBIC, MIT Photobook, Virage System and Columbia VisualSEEK and WebSEEK. A major drawback of the global histogram search is its sensitivity to intensity variations, color distortions and cropping.

In a human visual system, although color and texture are fundamental aspects of visual perceptions, human discernment of certain visual contents could potentially be associated with interesting classes of objects or semantic meanings of objects in the image. A region-based retrieval system segments images into regions (objects), and retrieves images based on the similarity between regions. If image segmentation is ideal, it is relatively easy for the system to identify objects in the image and to match similar objects from different images. Next, we review a CBIR system called SIMPLIcity (Semantics-sensitive Integrated Matching for Picture LIbraries) [15], which we use in our forensics experiments.

2.2 SIMPLIcity System

In the SIMPLIcity system, the query image and all database images are first segmented into regions. To segment an image, the system first partitions the image into non-overlapping blocks of size 4×4. A feature vector is then extracted for each block. The block size is chosen as a compromise between texture effectiveness and computation time. Smaller block sizes may preserve more texture details but increase the computation time. Conversely, larger block sizes reduce the computation time but lose texture information and increase segmentation coarseness.

Each feature vector consists of six features. Three of them are the average color components in a 4×4 block. The system uses the well-known LUV color space, where L encodes luminance, and U and V encode color information (chrominance). The other three represent energy in the high frequency bands of the wavelet transforms [3], i.e., the square root of the second-order moment of wavelet coefficients in high frequency bands.

To obtain these moments, a Daubechies-4 wavelet transform is applied to the L component of the image. After a one-level wavelet transform, a 4×4 block is decomposed into four frequency bands: LL (low low), LH (low high), HL and HH bands. Each band contains 2×2 coefficients. Without loss of generality, suppose the coefficients in the HL band are $\{c_{k,l}, c_{k,l+1}, c_{k+1,l}, c_{k+1,l+1}\}$. One feature is

$$f = \left(\frac{1}{4} \sum_{i=0}^{1} \sum_{j=0}^{1} {c_{k+i,l+j}}^2 \right)^{\frac{1}{2}} .$$

The other two features are computed similarly from the LH and HH bands. The motivation for using the features extracted from high frequency bands is that they reflect texture properties. The moments of wavelet coefficients in various frequency bands have been shown to be effective for representing texture. The intuition behind this is that coefficients in different frequency bands show variations in different directions. For example, the HL band shows activities in the horizontal direction. An image with vertical strips thus has high energy in the HL band and low energy in the LH band.

The k-means algorithm is used to cluster the feature vectors into several classes, each class corresponding to one region in the segmented image. Because clustering is performed in the feature space, blocks in each cluster do not necessarily form a connected region in the images. This way, segmentation preserves the natural clustering of objects in textured images and allows classification of textured images. The k-means algorithm does not specify how many clusters to choose. The system

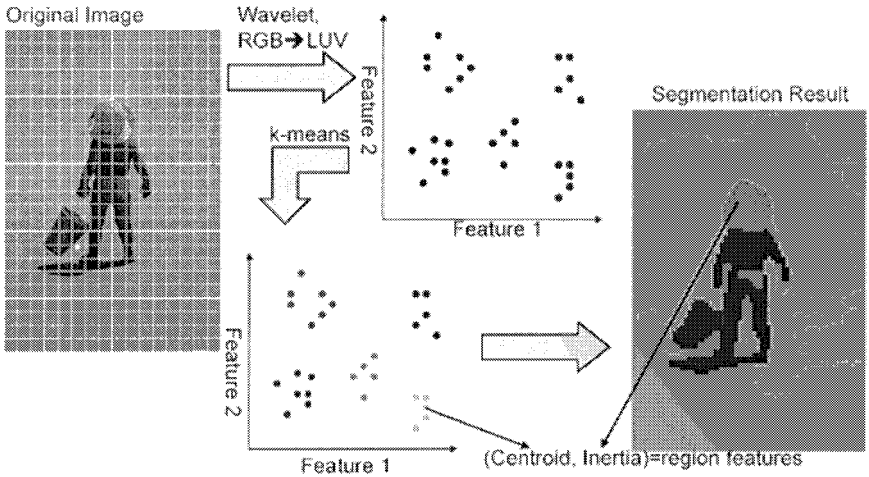


Figure 2. Schematic diagram of the feature extraction process.

adaptively selects the number of clusters, C, by gradually increasing C until a stopping criterion is met. The average number of clusters for all images in the database changes according to the termination criteria. Each region is represented by a feature vector (of dimension 6) that corresponds to the centroid of the cluster.

After segmentation, three extra features, normalized inertia [5] of orders 1 to 3, are calculated for each region to describe shape properties. The normalized inertia is invariant to scaling and rotation. The minimum normalized inertia is achieved by spheres. If an image is segmented into C regions, the image is represented by C feature vectors, each of dimension 9. Figure 2 illustrates the feature extraction process. Only two features for each image block are shown to make illustration easier. In the segmentation result, each region is represented by a distinct color.

The similarity between two images is computed according to an integrated region matching (IRM) scheme [8]. To reduce the influence of inaccurate segmentation, the IRM measure allows for matching a region of one image to several regions of another image (i.e., the region mapping between any two images is a many-to-many relationship). As a result, the similarity between two images is defined as the weighted sum of distances in the feature space, between all regions from different images. Compared with retrieval systems based on individual regions, the IRM approach decreases the impact of inaccurate segmentation by smoothing over the imprecision in distances.

3. Experimental Results

To evaluate the suitability of CBIR methods, we performed a number of experiments with the SIMPLIcity system. The experiments were designed to test its robustness against a number of typical transformed versions of the image that can be expected during an investigation. The first two were reductions in quality by varying the quality factor in JPEG images to 30% and 10%, respectively. Such variations can be expected for two reasons – to reduce storage requirements without noticeably impairing the visual perception (at screen resolution) and to provide (visibly) lower quality samples. Depending on the initial quality of the source images, the suitable values will vary. In our case, the vast majority of the pictures were taken with a 5 megapixel digital camera and we judged qualitatively that a quality value of 30% approximates the first scenario, whereas 10% approximates the second one.

Resizing is another common transformation applied for similar reasons, as well as to fit pictures into web pages. We tested three different versions at 512, 256 and 96 pixels (for the longer dimension) with the last one designed to simulate the common "thumbnailing" process. The last three transformations are 90 degree rotations and mirroring (vertical and horizontal) of images that can be expected during the processing of raw images.

The target database consisted of 5,631 photo images in JPEG format. The goal was to demonstrate the ability of the system to recognize an image when its altered version is submitted as the query. We applied image alteration to an image (called target image i) in the database. The resulting image i' is used as the query image and the rank of the retrieved target image i is recorded. The rank of image i is defined as the position of image i in the first 100 retrieved images. Clearly, a "good" system should return the original image at the top of the list (the best rank is 1). If image i does not show up in the top 100 retrieved images, it is considered a missed image.

We tested the system against the image alterations shown in Table 1. For each alteration, the average rank of all target images (excluding missed images) is computed; the results are given in Table 2. The experimental results indicate that image analysis techniques can significantly benefit digital forensic investigations. Of course, further study is warranted. Also, system-level issues such as performance, scalability and security must be addressed before a working prototype can be tested in a forensics laboratory. The following sections discuss the system design and the ongoing implementation effort.

Table 1. Alterations applied to query images.

ID	Alteration
JPEG30	Reducing JPEG quality to 10%
JPEG10	Reducing JPEG quality to 30%
Resize1	Resizing the image such that the largest of the width and height is 512 pixels
Resize2	Resizing the image such that the largest of the width and height is 256 pixels
Resize3	Resizing the image such that the largest of the width and height is 96 pixels
Rotation	Rotating the image by 90 degrees
Flip	Creating a mirror image
Flop	Creating a mirror image

Table 2. Experimental results for queries based on altered images.

Alteration ID	Missed Images (Miss Rate)	Average Rank
JPEG30	43(0.76%)	1.16
JPEG10	43(0.76%)	1.16
Resize1	43(0.76%)	1.16
Resize2	43(0.76%)	1.16
Resize3	43(0.76%)	1.16
Rotation	27(0.48%)	1.08
Flip	43(0.76%)	1.16
Flop	43(0.76%)	1.16

4. Design Goals

- *Service-Oriented Architecture*: The stored image feature sets are not contraband. However, even if they correspond to contraband images, we anticipate that the database will be administered by law enforcement agencies. Therefore, most software products will not be able to bundle such a database. Furthermore, the database will be a highly dynamic entity once a large number of law enforcement agencies become contributors.

- *Performance*: The system should be able to handle individual requests at rates that will allow investigations to proceed interactively. Current open source imaging software can generate thumbnail images at approximately 1000 images per minute using a single CPU. A working system should be able to perform at a similar rate or better (while providing a higher value service).

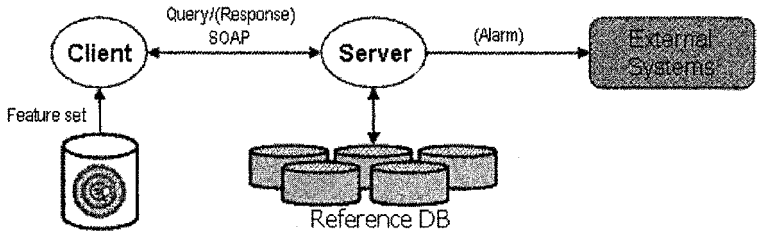


Figure 3. Architecture of a client/server image matching service. A client computes the feature set for one or more target images and issues queries to a server, which maintains a reference database of features for images of interest. The server indicates which images match the reference database and may alert external systems when matches occur.

- *Scalability*: The system should eventually be able to handle millions of images without a serious degradation in performance. This clearly implies that the system will have to incorporate replication and distributed processing as part of its original design.
- *Security*: The standard requirements for privacy, authentication and secure system administration apply. Recall that copies of the actual images are not stored. This makes the system legal and greatly mitigates the consequences of any security breach.
- *Flexible Deployment*: It should be possible to use the same architecture for forensics investigations and for preventive monitoring.

5. System Architecture

The architecture employs a client/server model (Figure 3). A client computes the feature sets of target images and submits them to the server. The server compares the submitted feature sets with those in a reference database. The client must keep track of outstanding queries and alert the user when image matches occur. Note that a match means that the feature set of the target image is close enough to a feature set in the reference database. Thus, false positives are a possibility, and these will have to be dealt with by the investigator.

The server has two basic functions:

1 Maintain the reference database of known feature sets. This includes adding and removing feature sets of images, as well as maintaining the integrity of the data and coordinating access to it. The latter two become non-trivial issues if the data and/or the processing are distributed for performance reasons.

2 Accept queries from clients and react accordingly. The server must first determine if the received feature set is a match, and then must either reply to the client, or take some other action, such as raising an alarm if the system is used for on-line monitoring.

The server is presented as a web service and uses a SOAP-based protocol to communicate with clients. The rationale is that the reference database is likely to be managed by a few law enforcement agencies and will have to be available over the Internet. The use of the public Internet is not a serious issue as a feature set is merely an array of numbers that are interpreted by the server. Standard mechanisms for authentication should still be in place to protect the database from attacks, e.g., denial of service. However, no unique security issues are raised by our design and, due to the nature of the database content, even a full-scale security breach will not yield any usable information.

Initial investigations of performance have confirmed that the processing of a forensic target will have to be distributed for it to be completed in a timely fashion. The dominant factor is the computation of image feature sets. Depending on source image size, this computation can take from a fraction of a second to a couple of minutes. In our work, images were scaled to not exceed 384×384 pixels, and the processing time was about 0.5 seconds per image. Since it would take about 14 hours to complete the feature extraction sequentially in a realistic scenario (100,000 images), we are attempting to integrate feature extraction into a distributed digital forensics infrastructure [13]. This infrastructure supports investigations on a cluster, providing vast improvements in performance over traditional "single investigator machine" approaches. The performance improvements arise from using multiple CPUs to tackle CPU-intensive operations and extensive caching to reduce disk I/O.

Another benefit of the distributed forensics infrastructure is that it could support case-specific searches on a target. Specifically, the system would build a reference database of all the images on the target and allow searches for images similar to the ones submitted interactively by an investigator, e.g., images containing a particular person or building.

6. Conclusions

This paper introduces a new approach for forensic investigations of visual images using content-based image retrieval (CBIR). The approach extracts an image "fingerprint" (feature set) and uses it to perform comparisons to find the best match among a set of images. It is necessary to store only the fingerprint (not the original image) to perform comparisons. The main advantage of the approach is that it allows the

construction of a reference database of fingerprints of contraband images. A secondary benefit is that it dramatically reduces the storage requirements for the reference database making it easier to achieve good performance at a reasonable cost.

Experiments indicate that CBIR techniques are well-suited for forensic purposes. In particular, the tests of robustness of query results for versions of the original images obtained through common transformations (e.g., resizing) are very promising.

Two main applications are proposed: a reference database for contraband images and case-specific image search tools. In the first case, law enforcement agencies will be able to collectively build and access the database to automatically search targets for known contraband images. In the second case, a database of all images found on a target is constructed and investigators can submit queries for images similar to specific images of interest. To accommodate these applications, a service-oriented architecture and a distributed forensic tool are proposed.

The main contribution of this work is that it presents a sound and practical approach to automating the forensic examination of images. Unlike other approaches, such as hashing, the image analysis approach is very stable in that it can locate not only the original image but also common variations of the image.

Acknowledgments

The research of Yixin Chen was supported in part by NASA EPSCoR Grant NASA/LEQSF(2004)-DART-12 and by the Research Institute for Children, Children's Hospital, New Orleans, Louisiana. The authors also wish to thank James Z. Wang and Jia Li for providing executable code of the SIMPLIcity System.

References

[1] C. Carson, S. Belongie, H. Greenspan and J. Malik, Blobworld: Image segmentation using expectation-maximization and its application to image querying, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24(8), pp. 1026-1038, 2002.

[2] I. Cox, M. Miller, T. Minka, T. Papathomas and P. Yianilos, The Bayesian image retrieval system PicHunter: Theory, implementation and psychophysical experiments, *IEEE Transactions on Image Processing*, vol. 9(1), pp. 20-37, 2000.

[3] I. Daubechies, *Ten Lectures on Wavelets*, Capital City Press, Philadelphia, Pennsylvania, 1992.

[4] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic and W. Equitz, Efficient and effective querying by image content, *Journal of Intelligent Information Systems*, vol. 3(3-4), pp. 231-262, 1994.

[5] A. Gersho, Asymptotically optimum block quantization, *IEEE Transactions on Information Theory*, vol. 25(4), pp. 373-380, 1979.

[6] T. Gevers and A. Smeulders, PicToSeek: Combining color and shape invariant features for image retrieval, *IEEE Transactions on Image Processing*, vol. 9(1), pp. 102-119, 2000.

[7] A. Gupta and R. Jain, Visual information retrieval, *Communications of the ACM*, vol. 40(5), pp. 70-79, 1997.

[8] J. Li, J. Wang and G. Wiederhold, IRM: Integrated region matching for image retrieval, *Proceedings of the ACM International Conference on Multimedia*, pp. 147-156, 2000.

[9] W. Ma and B. Manjunath, NeTra: A toolbox for navigating large image databases, *Proceedings of the IEEE International Conference on Image Processing*, pp. 568-571, 1997.

[10] S. Mehrotra, Y. Rui, M. Ortega-Binderberger and T. Huang, Supporting content-based queries over images in MARS, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 632-633, 1997.

[11] V. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images, *IEEE Computer*, vol. 28(9), pp. 40-48, 1995.

[12] A. Pentland, R. Picard and S. Sclaroff, Photobook: Content-based manipulation for image databases, *International Journal of Computer Vision*, vol. 18(3), pp. 233-254, 1996.

[13] V. Roussev and G. Richard III, Breaking the performance wall: The case for distributed digital forensics, *Proceedings of the Digital Forensics Research Workshop*, 2004.

[14] J. Smith and S. Chang, VisualSEEK: A fully automated content-based query system, *Proceedings of the ACM International Conference on Multimedia*, pp. 87-98, 1996.

[15] J. Wang, J. Li and G. Wiederhold, SIMPLIcity: Semantics-sensitive integrated matching for picture libraries, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23(9), pp. 947-963, 2001.

[16] J. Wang, G. Wiederhold, O. Firschein and X. Sha, Content-based image indexing and searching using Daubechies' wavelets, *International Journal on Digital Libraries*, vol. 1(4), pp. 311-328, 1998.

Chapter 23

MAKING DECISIONS ABOUT LEGAL RESPONSES TO CYBER ATTACKS

L. Peng, T. Wingfield, D. Wijesekera, E. Frye, R. Jackson and J. Michael

Abstract Cyber intrusions may be characterized in one or more of three legal regimes: law enforcement, intelligence collection and military operations. Furthermore, most intrusions occur across a number of jurisdictional boundaries, building complex conflict-of-laws questions into such attacks. Applying a one-size-fits-all response, such as always terminating all interaction with the intruder or always responding in kind, can be an ineffective or even worse, illegal, response. In order to assist investigators and legal experts addressing the legal aspects of cyber incidents, we have developed a decision support tool that takes them through a series of questions that are akin to those posed by an attorney to a client seeking legal guidance. Our tool may be used by builders and users. Builders use the tool to construct trees of legal arguments applied to the incidents at hand with the documentation useful for building legal briefs. Users interact with the tool by answering a series of questions to obtain viable legal arguments with supporting documents.

Keywords: Cyber attacks, legal issues, decision support system, incident response

1. Introduction

When a malefactor intrudes into a computer system, the owner of that system – whether a private individual protecting personal property, a corporation securing its assets, or a government defending its interests – needs to know something about the malefactor in order to develop a lawful and effective response to the intrusion. Cyber intrusions may be characterized in one or more of three legal regimes: law enforcement, intelligence collection and military operations. Furthermore, intrusions can occur across a number of jurisdictional boundaries, building complex conflict-of-laws questions into such attacks. Applying a one-size-fits-all response, such as always terminating all interaction with the intruder or

always responding in kind, can be an ineffective or even worse, illegal, response. For instance, terminating interaction with an intruder could prevent the seizure of evidence for criminal prosecution, collection of information for counterintelligence purposes, or counter-targeting for a military response [9]. By responding in kind, the defender may violate domestic or international law, or, in the case of a government actor, inadvertently escalate to the level of a use of force or even an armed attack. Furthermore, an inappropriately calibrated response may contravene the customary rules of war accepted as authoritative law by the United States – distinction, necessity, proportionality and chivalry.

The general problem we address in this paper is that of providing defenders with sufficient information to make informed decisions when formulating responses to intruders. Specifically, we describe a tool that serves as an automated aid for determining the legal regime under which a cyber intrusion can be categorized, with all documentation supporting the building of a brief. Our tool is built on the premise that owners and their agents of affected computing resources want to defend their computer systems without violating domestic or international law.

Both the frequency and intensity of attacks in cyberspace can be high, affording little time for research and thoughtful consideration before the cyber intrusion (whether a theft or an attack) is over. Similarly, what may initially appear to be a minor intrusion or misuse of a computer system may ultimately result in damage to or destruction of property, or even human injury or loss of life. In either case, the defender must be prepared to respond to such attacks with operational plans and mechanisms for real-time information collection already in place. In other words, the defender needs to tighten his Observe-Orient-Decide-Act (OODA) loop in order to gain a competitive advantage over the intruder [7].

Legal preparation is an essential element in this equation. Against opponents who disregard any laws which are not immediately and effectively punitive, the default response of inadequately counseled operators is to forego otherwise lawful and effective defensive strategies. In other words, the vast legal gray area that exists today operates in favor of the intruder – a form of asymmetry between the attacker and the defender. A clearer and more timely picture of the operational legalities of the situation would provide the defender with more, rather than fewer, options. At this stage in our research, several caveats are in order. First, the present tool is illustrative of the concept, and is not intended to be employed operationally at this point. The questions and answers have an artificially academic clarity, which derives from top-down reasoning of broad questions to narrow circumstances. Second, the decision-tree format no longer defines the state-of-the art in

expert systems, but it does: (a) present the core concepts clearly, (b) provide a framework that clarifies the transparent assembly of resources supporting legal analyses, and (c) lay a foundation for more elaborate logical structures (such as totality-of-the-circumstances analysis) for future operational employment. Third, the inevitable anomalies which will arise in its development (i.e., requiring an early legal determination of whether or not the intruder is a US person will almost certainly conflict with the operational reality of discovering key facts late in the game) serve to highlight conflicts and lacunae in the law. The degree to which the most operationally useful flow of legal questions fails to meet real world requirements is the degree to which the law or technology must change. Fourth, this tool will be developed in alignment with international law, but numerous questions (especially in the law enforcement and intelligence collection realms) will never rise to the level of state vs. state legal determinations. Where national and international law appear to conflict, that tension will be made explicit and thus clarified for resolution.

The remainder of this paper is organized as follows. Section 2 describes the details of our application requirements. Section 3 describes the software design of our toolkit. Section 4 explains the functionality of the tool through an example. Section 5 describes related work and Section 6 concludes the paper.

2. Legal Requirements

As stated, our objective is to enforce legal responses to cyber incidents. In doing so, we are guided by the legal advice given to litigants who claim that they have a legitimate case for recourse. When a litigant discusses his or her situation with an attorney or investigator, the latter asks a series of questions to determine the applicable legal regime and to map out a course of action. Our larger objective is to make this a primary global requirement for responding to incidents in a timely manner. To reason about response alternatives, we first need a model of the domestic and international law governing cyber intrusions, one for computers to execute without the human in the loop and at high speed, and another for human decision making at considerably lower speed. Our proposal for this model is a customizable decision tree of legally relevant questions, modeling those that would be asked by an investigator from a prospective complainant. While the computer's decision capability encoded in the form of a tree can be hardwired for independent execution of clearly discernable, objectively verifiable criteria, the decision tree to be manually traversed will have pre-selected sources available to assist the

attorney in deciding each of the gray area judgments requiring human reflection and creativity. It is necessary to assemble a comprehensive selection of sources to append these to each decision point, but it will be vital, for speed and clarity, to include no more than is required to answer the question at hand. These sources may be grouped as constitutional, legislative (statutes), executive (regulations), judiciary (cases) and international. These five categories must be further subdivided into primary (the case or statute itself), and secondary (analytic and synthetic commentary, such as law review articles, or briefs on file). These ten categories are sufficient to contain any legal source needed to address any given question. Furthermore, each source would have to be presented at four levels of abstraction, for the proper balance of speed and depth:

- Citation: A legal footnote.
- Précis: A sentence or paragraph paraphrasing what the source has to say about the question at hand.
- Excerpt: Direct quotes from the source which are on point
- Document: The complete law review article, statute or case.

This general information would be distilled into a specific research question in two media: an audit trail, providing a record of each question asked and each answer chosen, and a brief builder, which would augment the audit trail with those portions of the sources selected by the reviewing attorney to support his answer to the question. This would, in effect, be the first draft of a legal brief supporting the selected course of action.

The decision tree, and its supporting sources, may be constructed using an open source methodology, allowing law students, practitioners, and scholars scattered across the world to collaborate on its construction and refinement. With the process architecture (described below) in place, the trees will be available to selected legal academics for analysis and improvement. Designing such a legal analysis tool for a comprehensive tracking system will be of great benefit to the cyber-legal community, because it will require the analysis and distillation of the entire field into the simplest possible framework for implementation.

This system will take the form of a set of predefined sequential questions when an actor's behavior indicates he or she may be intruding into, misusing or attacking a computer system. To simplify the logic employed, in the prototype, each question has only *yes* and *no* answers. A deferent question will follow each *yes* or *no* answer to continue the

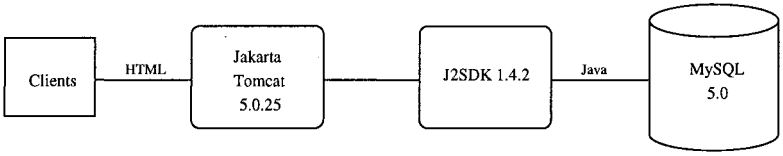


Figure 1. System architecture.

analysis. Then attorneys and their clients would follow a complete logical path to reach a transparently reasoned legal conclusion. A third option, *don't know* allows the user to view the legal resources necessary to proceed forward with a *yes* or *no* answer. As mentioned earlier, these resources are arrayed in ten categories (constitutional, legislative, executive, judicial and international, each at a primary and secondary level), and each source may be accessed at any one of four levels of abstraction (citation, précis, quotation and full source).

The system will operate on two levels: for users following previously constructed analyses, and for builders, assembling and testing the analyses to be provided to the operational community. Users are attorneys responsible for providing operational legal advice to law enforcement, intelligence community, or military officials. These users follow a decision tree, answering a sequence of questions carefully crafted to identify and record the legally operative facts of the incident. This decision support tool will produce a logical legal analysis, supported by the legal resources selected by the user. Builders are academic or practicing attorneys and some computer network technicians, adding and subtracting branches from the decision tree and the resources available at each decision point. They create and maintain the substance of the decision support tool.

3. System Design

The prototype is designed to be an open-source, web-enabled decision support tool that provides legal reasoning web services. Multiple clients may access the web server (the system) via web browsers, such as Internet Explorer or Netscape. The communication language between clients and web server is HTML exposed within Java Server Pages (JSP). A Java engine, Java 2 Software Development Kit (J2SDK), is used to compile JSP pages to Java class files that send a HTML stream to web clients and communicate with a mySQL database through JConnector technology. Figure 1 shows the system architecture. Compared to client-server applications, among others, this multi-tier design has the following advantages:

- Clients may remotely and concurrently access the system, sharing the same knowledge base.
- The architecture is extensible, because it is built using Java 2 Enterprise Edition (J2EE) service framework, with quick deployment times and minimal maintenance efforts in mind.
- The system can be extended to use RDF, OWL, RuleML or JESS as needed.
- The system is easily manageable, because some clients are allowed to change the knowledge base while other clients can only access built-in scenarios.

Each client (actor in software engineering) is a builder or user with his/her own separate applications that share one database and file system. User functionality includes answering questions, getting a decision, viewing (audit trail, tree map, legal brief), searching pertinent legal documents, and displaying legal documents. Builder functionality includes adding and deleting trees/decisions/questions, linking decisions/resources and loading resources.

3.1 Detailed User Requirements

- The system should collect legally relevant facts
- The system should follow the decision tree, answering *yes, no* or *don't know* to each question in sequence. A user should be able to go back to prior questions and change answers to evaluate the consequences of alternative answers. An *yes* or *no* answer proceeds to the next (pre-determined) question. While many possible paths are available, any given sequence of *yes* or *no* answers should yield only one result.
- For a *don't know* answer, the system should present legal resources to assist in making a *yes* or *no* decision. These resources will vary in number and length depending on the question at hand, but are grouped by category (constitutional, legislative, executive, judicial and international), each with a primary and secondary set of materials, and subcategories such as country and language. Each resource is accessible to four levels of detail: citation, précis, excerpt and source.
- The audit trail function should display the history of navigation with consulted sources in the citation format along with answers provided by the user.

- The brief builder function should do the same, and include all user-selected portions of consulted sources.
- After a sufficient number of questions have been answered, the system should provide a decision with supporting documents. A user should be able to search databases uploaded into the tool under the ten categories.
- The system should display searched resources at four levels of detail (citation, pécis, excerpt and source).

3.2 Detailed Builder Requirements

- Builders should be able to login and navigate any of the tool's web pages, including those of users. For security reasons, builders inactive for thirty minutes are logged out.
- After initiating a new decision tree or selecting an existing one, a builder should be able to add any answer *(yes/no/don't know)* and link either of these to another question.
- Builders should be able to upload relevant documents and categorize them in support of the *don't know* option.
- Builders should be able to separate each resource into its appropriate levels of abstraction (citation, pécis, excerpt and source).
- Builders should be able to delete a question, a decision or an entire tree.

4. System Functionality

This section describes the functionality of the tool by constructing an example decision tree to determine the answer to the legal question:*Are we at war?* (see Figure 2). The builder can access the system to build a decision tree via a web browser after a correct login as shown in Figure 3.

As a first step, the builder creates a new tree named: *Are we at war?* Then, the builder adds three possible decisions to this new tree. Next, the builder inserts multiple questions and links the right follow-up questions or decisions with them. The builder needs to specify the parent question in the tree that to which the new question is to be linked; that is, the builder should design the system so that a *yes* or *no* answer to a previous question is linked to a new question posed to a user as shown in Figure 4. Because decision trees can be complex, the system is designed to offer the builder flexibility. For example, the builder can input the system's decision and questions without having to enter the links when

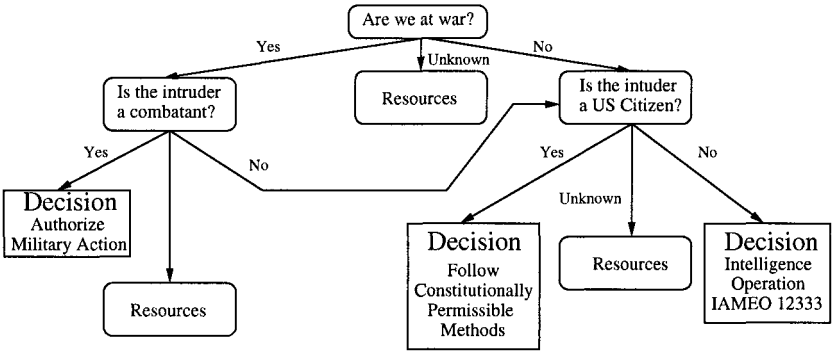


Figure 2. Decision tree.

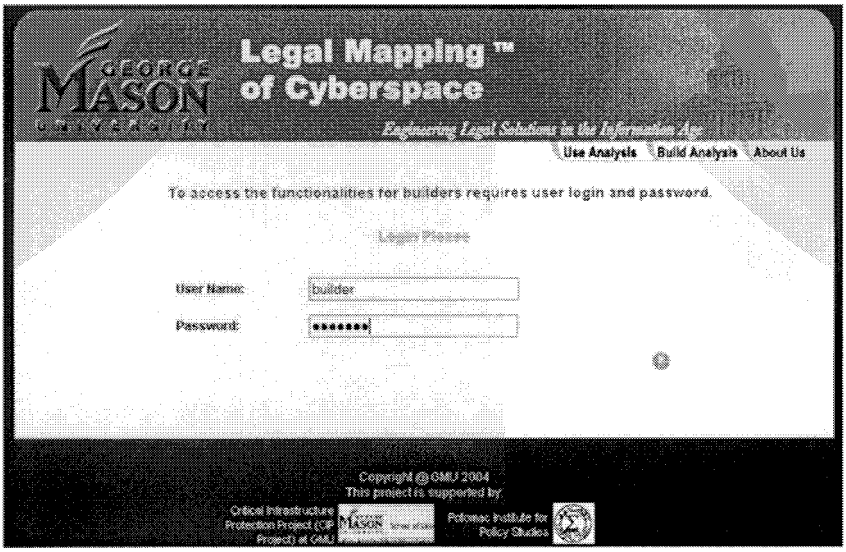


Figure 3. Login page of the tool.

specifying them. After that, the builder can use menu options to link the decisions and questions. The complete decision tree can be constructed in multiple ways as shown in Figure 5.

5. Related Work

Although there have been numerous academic attempts to elicit a logical structure from legal decision-making processes, none is in widespread use with practicing attorneys. The proprietary databases of Westlaw and Lexis-Nexis, searchable by document category and Boolean keyword strings, are the most frequently consulted by attorneys. Both have an

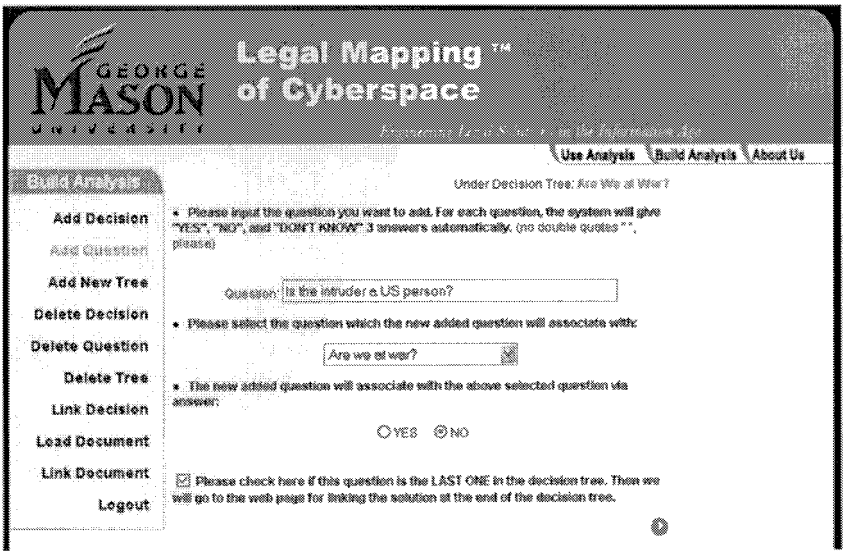


Figure 4. Linking questions.

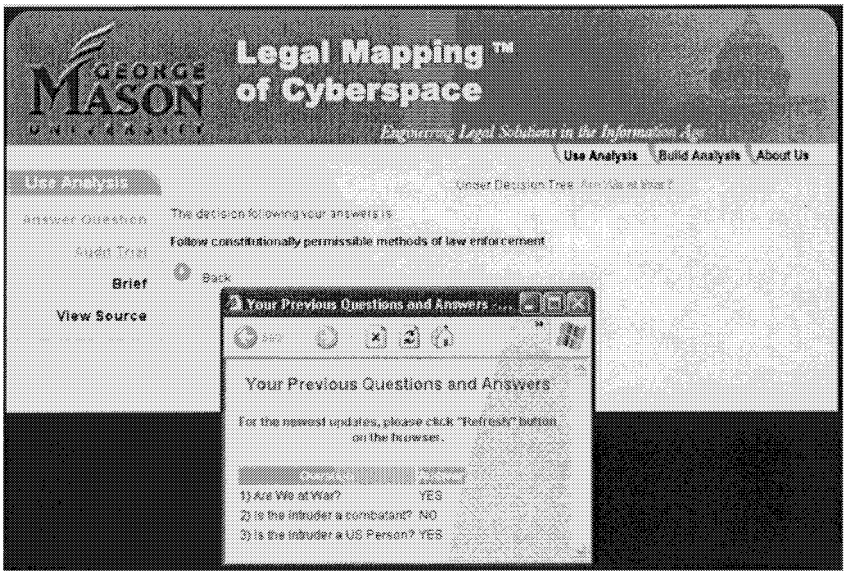


Figure 5. Decision rendered by a completed tree.

impressive number of cases, briefs, law review articles, and related documents [4, 5, 8], but neither is intended to provide direct assistance with the formulation or execution of a legal analysis. Furthermore, there are numerous *free* sites on the Internet – mostly maintained by universities – that have large searchable databases. Like their commercial analogs, they provide quick and reliable access to the documents themselves, but are not designed to assist in legal analyses *per se*. The University of Minnesota's Human Rights Library is an excellent example of such a system; it is the source of U.N. Charter text provided in one of our examples [10].

Capturing legal knowledge and enabling legal discourse is technically challenging and a continuous effort because laws and their interpretations change over time. Several legal reasoning support tools, e.g., [11, 12, 13], are used primarily by law students to hone their analytical skills. Others are geared for methodology or ontology research [3, 6, 13]. Only a few of these are complete web-based tools used for general legal reasoning [1, 14, 15], and therefore are not specific to one area of law. Digud [2] and Zeleznikow [16] have developed web-based tools for reasoning about divorce laws and enhancing access to legal process. In contrast, our tool may be used to train law students and cyberspace technicians, as well as to provide legal support for responding to cyber intrusions. Being both web-based and open source increases its usability, extensibility, maintainability and potential for incremental enhancements.

6. Conclusions

Due to the need to keep responses to cyber attacks legal, responders need to be aware of the legal support available within a given legal framework. To address this need, we have developed a decision-tree-based tool that takes potential investigators and attorneys through a series of questions to help build legal briefs against perpetrators. In order to do so, the decisions have to be constructed by attorneys who are well versed in this area of law: they construct trees of sequentially-ordered questions that guide users through to an actionable recommendations for response (i.e., answers presented at terminal leaves in a tree). In addition, our toolkit stores relevant information within legal categories (e.g., constitutional, legal, international) at four levels of detail (footnote, précis, excerpt, entire document) necessary to build legal briefs.

To improve the usefulness of our tool and serve a diverse community of international users, we are in the process of populating it with legal documents related to critical infrastructure protection from different countries. Because different legal systems use different ontologies,

we are designing an interoperable abstract super-ontology that specializes to different legal systems. The super-ontology facilitates semantic searching across legal ontologies in addition to introducing a degree of transparency in that the user does not need to be familiar with a foreign legal system. For the aforementioned purpose, we follow a two-pronged approach. For the short term, we analyze international legal documents and categorize their legal discourse ontologies. For the long term, we are developing an abstract ontology that can be adapted to the specific doctrines that are used in various areas of the law.

References

[1] K. Curran and L. Higgins, A legal information retrieval system, *Journal of Information, Law and Technology*, vol. 3, 2000.

[2] S. Duguid, L. Edwards and J. Kingston, A web-based decision support system for divorce lawyers, *Journal of Law, Computers and Technology*, vol. 15, pp. 265-280, 2001.

[3] A. Gangemi, A. Prisco, M. Sagri, G. Steve and D. Tiscornia, Some ontological tools to support legal regulatory compliance, *Proceedings of the Workshop on Regulatory Ontologies and the Modeling of Complaint Regulations*, *Lecture Notes in Computer Science (Vol. 2889)*, Springer-Verlag, Berlin Heidelberg, Germany, pp. 607-620, 2003.

[4] C. Hafner, Legal reasoning models, in *International Encyclopedia of the Social and Behavioral Sciences*, Elsevier, Amsterdam, The Netherlands, 2001.

[5] C. Hafner and D. Berman, The role of context in case-based legal reasoning: Teleological, temporal and procedural, *Artificial Intelligence and Law*, vol. 10, pp. 19-64, 2002.

[6] M. Hall, A. Stranieri and J. Zeleznikow, A strategy for evaluating web-based decision support systems, *Proceedings of the Sixth East-European Conference on Advances in Data Information Systems*, 2002.

[7] P. Huygen, Use of Bayesian belief networks in legal reasoning, *Proceedings of the Seventeenth British and Irish Legal Education Technology Association Conference*, 2002.

[8] E. Katsh and J. Rifkin, *Online Dispute Resolution: Resolving Conflicts in Cyberspace*, Jossey-Bass, San Francisco, California, 2001.

[9] J. Michael, On the response policy of software decoys: Conducting software-based deception in the cyber battlespace, *Proceedings of the Twenty-Sixth Computer Software and Applications Conference*, pp. 957-962, 2002.

[10] J. Michael and T. Wingfield, Lawful cyber decoy policy, in *Security and Privacy in the Age of Uncertainty*, D. Gritzalis, *et al.*, (Eds.), Kluwer, Boston, Massachusetts, pp. 483-488, 2003.

[11] A. Muntjewerff, Automated training of legal reasoning, *Proceedings of the Ninth British and Irish Legal Education Technology Association Conference*, pp. 51-58, 1994.

[12] A. Muntjewerff, A. Jordaans, R. Huekstra and R. Leenes, Case analysis and storage environment (case), *JURIX*, 2002.

[13] V. Randall, Online academic assistance for law students (academic.udayton.edu/legaled/online/).

[14] A. Stranieri, J. Yearwood and J. Zeleznikow, Tools for placing legal decision support systems on the World-Wide Web, *Proceedings of the Eighth International Conference on Artificial Intelligence and Law*, pp. 206-214, 2001.

[15] A. Stranieri and J. Zeleznikow, Tools for intelligent decision support system development in the legal domain, *Proceedings of the Twelfth IEEE International Conference on Tools with Artificial Intelligence*, pp. 186-189, 2000.

[16] J. Zeleznikow, Using web-based legal decision support systems to improve access to justice, *Journal of Information and Communication Technology Law*, 2002.

Chapter 24

APPLYING FILTER CLUSTERS TO REDUCE SEARCH STATE SPACE

Jill Slay and Kris Jorgensen

Abstract Computer forensic tools must be both accurate and reliable so as not to miss vital evidence. While many investigations are conducted in sophisticated digital forensic laboratories, there is an increasing need to develop tools and techniques that could permit preliminary investigations to be carried out in the field. Pre-filtering electronic data in the field, before a computer is brought back to a laboratory for full investigation, can save valuable time. Filtering can also speed up in-house investigations by reducing search space size.

This paper discusses the application of automated tools based on filters. In addition to helping reduce the search space, filters can support specific tasks such as locating and identifying encryption software and hidden, encrypted or compressed files. Filters may be used to automate tedious examinations of temporary Internet files, Windows directories or illicit images. Also, filters can facilitate customized searches based on patterns encountered in investigations of common cases.

Keywords: Forensic tools, field investigations, filter design, filter clusters

1. Introduction

One of the main challenges in computer forensic investigations is the increasing capacity of storage media [1, 7]. Even small electronic devices can hold thousands of documents, images and other files. Because of the capacity of these devices, it is necessary to devise techniques that can partition the search space into smaller, more easily managed areas. Partitioning the search space makes it more feasible to locate specific data, whether or not attempts have been made to conceal it.

Extracting digital evidence from storage devices requires the use of forensically sound tools and techniques [7, 8]. In contrast to normal applications where computers are used to significantly decrease the time

needed to reach an objective, the priority in a computer forensic investigation is accuracy rather than speed [3]. It is often impractical to do a complete examination of all the devices encountered in an investigation. This problem is further aggravated by rapid increases in storage capacity [4]. Moreover, achieving the desired level of accuracy is particularly difficult given the exigencies in field investigations.

Several techniques exist for concealing data in electronic devices, ranging from simple techniques such as hiding a file in a large collection of other files to advanced techniques involving alternate data streams, encryption and steganography [2]. Given the availability of free software for data concealment, it is a distinct possibility that incriminating information may have a high level of protection applied to it.

A forensic examiner must not only have the tools for conducting investigations, but must also have a solid grounding in and knowledge of operating systems, file systems, file formats and information storage techniques [9]. With constantly changing technology, examiners must be kept up to date by training or self-learning. Even then, it is not possible for one individual to be an expert in all areas. This highlights the need for a team approach, with each member of the team having expertise in specific areas.

Forensic tools are available for performing tasks such as searching for file types, detecting encryption, recovering deleted files, locating concealed data and tracing email. Some advanced tools allow scripts to be written to tailor their operations to specific investigations.

While many investigations are conducted in sophisticated digital forensic laboratories, there is an increasing need to develop tools and techniques that could permit preliminary investigations to be carried out in the field, especially in remote locations and rural areas. Pre-filtering electronic data in the field, before a computer is brought back to a laboratory for full investigation, can save valuable time. Filtering can also speed up in-house investigations by reducing search space size.

This paper discusses the application of automated tools based on filters. In addition to helping reduce the search space, filters can support specific tasks such as locating and identifying encryption software, and hidden, encrypted or compressed files. Filters may be used to automate tedious examinations of temporary Internet files, Windows directories or illicit images. Also, filters can facilitate customized searches based on patterns encountered in investigations of common cases.

2. Exhaustive Search vs. Intelligent Search

Due to the massive capacity of modern storage media, the actual amount of incriminating data in many cases is a small percentage of the total data contained on seized media. Two approaches exist for locating evidence in a mass storage device, the "Tourist Approach" and the "Divide and Conquer Approach."

The Tourist Approach involves examining every bit of data on the device. While this approach is very time consuming and is often infeasible [12], notwithstanding human error, it will yield the data sought if such data, in fact, exists on the device. Of course, it may not be possible recover data that is protected using cryptography or steganography [5].

The Divide and Conquer Approach applies intelligence to partition the data into pre-established domains that are analyzed individually. The partitioning process does not, and probably should not, entail just one pass as further partitions could be sought that logically group the data into the smallest sets possible. Once the data is sorted into logical groupings, more specific approaches can be applied to each data set. Examples include automatically detecting images based on the presence of skin-colored pixels or culling files that contain specific keywords and phrases. By using a collection of divide-and-conquer partitioning strategies, the original data can be sifted automatically, enabling an examiner to significantly reduce search time by focusing on specific data sets.

3. Determining What to Exclude

One method to reduce the search space is to determine the data that is superfluous to the investigation, and eliminate this incidental data at the outset.

The level of skill of a suspect may decide the extent to which data is eliminated. The skill levels, and therefore the types of filters, range from a barely computer-literate user to an expert who can hide data in operating system files without impeding normal system use. In the case of a suspect with average computer skills, an investigator could ignore operating system files and scan user-created files. This can be accomplished by using filters designed to "bubble up" user data that is relevant to the investigation.

Often, the only data that actually pertains to a case is the data that the user has explicitly placed on the storage device. Most of the other incidental data can be safely eliminated without fear of ignoring crucial evidence. A prime example is incidental data placed on the device by the operating system to permit its operation. For example, in the case of

Microsoft Windows, the system dependent DLLs and executables need not be considered and can be eliminated during an initial search.

Some individuals may be competent enough to conceal data so that it cannot be found except by exceptional means, e.g., using second harmonic magneto resistive microscopy [10]. Therefore, all storage media from suspects with technical expertise should be immediately seized and sent to a laboratory for a complete investigation

During a thorough laboratory-based forensic investigation, the process of elimination of inconsequential files is done in a controlled manner. Toolkits utilizing hash sets allow for low-level matching of files to known signatures [11]. These hash sets are available from commercial vendors, law enforcement agencies and government organizations, e.g., the U.S. National Institute of Standards and Technology [11].

4. Designing Filters

To make the process of locating relevant information more efficient, tools must be developed that can narrow the search state space of a device being scrutinized. These tools are the digital forensic equivalent of filters in the real world.

A filter partitions data based on specific criteria. By combining multiple filters into filter clusters, it is possible to narrow down the target device information and attach a relevant suspicion level to data, which in effect ranks the data from most likely to be relevant to least likely to be relevant. Filters should be used in order of increasing specificity: the first filter should partition data as useful and not useful, and remaining filters should become more specific as the desired outcome is approached.

Filters need not be designed to produce wide partitions of the search space. Instead, filters can target individual files or programs, eliminating the need to have different search programs or strategies. In these instances, however, the filter loses its ability to be included as part of a logical formula. This is not necessarily a bad option as it precludes the use of more than one tool and allows the same design paradigm to be applied to search one specific case or to divide the space for other filters or human examination.

No two suspects are likely to hide data in exactly the same way. Therefore, given the myriad ways available for concealing data, a single filter type cannot handle every case. Instead, multiple filters should applied in series, each designed to give priority to a different aspect. This implies an iterative development process in which filters that are found to be effective are added to the toolkit. Also, if one filter cluster fails to provide meaningful results, others can be applied to the data. If the

filter clusters used in a field investigation do not provide useful information, the target device should be sent to a laboratory for a complete examination. Data recovered during the examination will have the side effect of producing useful filter clusters for future cases.

5. Filter Types

Filters are divided into four main categories based on logical principles [6]: inclusion, exclusion, grouped and isolated filters. By reducing the filter types to logical operations and using set theoretical properties, it is possible to strictly define filters in terms of their purpose.

- **Inclusion Filters:** These filters are defined in terms of information that should be included in the result, e.g., all JPEG files on a device.
- **Exclusion Filters:** These filters are defined in terms of information that should be excluded from the result, e.g., exclude all images smaller than 50mm $\times$ 50mm.
- **Grouped Filters:** These filters specifically target similar types of data in the general vicinity of each other. Depending on how the filter is specified, it could select files in the same directory, or files in neighboring directories, or 90% of the same type of files in the same directory or in neighboring directories.
- **Isolated Filters:** These filters specifically target dissimilar types of data in the general vicinity of each other. An example is a small number of files of one type that are dispersed among a large number of files of different types.

Intersection is a useful operation for combining the results of different filters. The intersection operation on filters yields four possible outcomes: (i) Accept everything from both filters, (ii) Accept all but the intersection, (iii) Accept the intersection, and (iv) Accept the difference.

A collection of filters can be used in different combinations to implement complex selection criteria similar to using logical formulas. Individual filters can also be organized into clusters, resulting in successive layers of inclusion and exclusion. The selection criteria of the initial filter determine the data that is operated on by the remaining filters. Data that is not within the filter's criteria is placed in the set of non-processed data. This set is of interest because it may contain concealed, malformed, unknown and/or deleted data.

The purpose of a forensic examination is to locate incriminating data. Therefore, it stands to reason that a suspect may have gone to some

trouble to deliberately hide incriminating data. If this is the case, the data being sought could fall into the excluded set and a search for concealed data should concentrate on this data set.

Collections of filters can be specified based on case experience. These filters may be embedded into automated tools designed for field investigations. By validating these tools against the requirements of forensic analyses, it is possible to ensure the authenticity of the recovered data.

6. Conclusions

Filters help reduce the size of the search state space in forensic investigations of storage media that hold large amounts of data. Pre-filtering electronic data in the field or during an in-house investigation can save valuable time. In addition, filters support specific tasks such as locating and identifying encryption software and hidden, encrypted or compressed files. Filters can automate tedious examinations of temporary Internet files, Windows directories or illicit images. Filters also facilitate customized searches based on patterns encountered in investigations of common cases. Finally, filters can help collect statistical information on common data hiding techniques and effective search strategies.

References

[1] M. Anderson, Hard disk drives: Bigger is not better (www.forensics-intl.com/art14.html), 2001.

[2] AntiOnline.com, Basic data hiding tutorial (www.antionline.com /printthread.php?threadid=251463&pagenumber=1), 2004.

[3] DIBS, The DIBS Methodology (www.dibsusa.com/methodology /methodology.html).

[4] M. Hannan and P. Turner, Australian forensic computing investigation teams: Research on competence, *Proceedings of the Seventh Pacific-Asia Conference on Information Systems*, 2003.

[5] ITsecurity.com, Encryption (www.itsecurity.com/security.htm?s=386), 2004.

[6] R. Johnsonbaugh, *Discrete Mathematics*, Prentice Hall, Englewood Cliffs, New Jersey, 2001.

[7] W. Kruse and J. Heiser, What exactly is computer forensics? (www.developer.com/java/other/article.php/3308361.01), 2004.

[8] R. McKemmish, What is forensic computing? *Australian Institute of Criminology: Trends & Issues in Crime and Criminal Justice*, pp. 1-6 (www.cit.uws.edu.au/compsci/computerforensics/Online%20Materials/ti118.pdf), 1999.

[9] D. Michaud, Adventures in computer forensics (www.sans.org/rr /papers/27/638.pdf), 2001.

[10] NIST, New Commerce Department magnetic microscope helps retrieve information from damaged or altered tapes (www.nist.gov /public/_affairs/releases/g00-108.htm), 2001.

[11] NIST, National Software Reference Library and Computer Forensics Tool Testing Project (www.nsrl.nist.gov/Project), 2003.

[12] M. Noblett, M. Pollitt and L. Presley, Recovering and examining computer forensic evidence, *Forensic Science Communications,* vol. 2(4), 2000.

Chapter 25

IN-KERNEL CRYPTOGRAPHIC EXECUTABLE VERIFICATION

Yusuf Motara and Barry Irwin

Abstract This paper discusses the problems posed by Trojan horses and unauthorized code, and reviews existing solutions for dealing with them. A technique involving the in-kernel verification of executables is proposed. Its advantages include simplicity, transparency, ease of use and minimal setup time. In addition, the technique has several applications, including assisting with honeypot implementations, incident response and forensic investigations.

Keywords: Trojan horse, signed binaries, executable verification, cryptography

1. Introduction

Computer users around the world are continuously bombarded with viruses, worms and exploits, many of which leave unwanted executable content on their machines. Current solutions to the problem include patching, using anti-virus programs or securing systems with firewalls, all of which are preventative measures [10]. No standard technique exists to stop a program from executing if it manages to bypass security controls. This paper describes a technique involving the in-kernel verification of executables, which detects unauthorized code and (optionally) prevents it from executing.

The following section discusses the problems posed by Trojan horses and unauthorized code, and reviews existing solutions for dealing with them. Next, the technique involving in-kernel verification of executables is described in detail. Finally, applications of the technique, including honeypot implementations, incident response and forensic investigations, are highlighted.

2. Background

This section discusses Trojan horses and unauthorized code, and evaluates techniques for guarding against them involving integrity checking, pre-execution validation and comprehensive security solutions.

2.1 Trojan Horses

A Trojan horse is defined as "a malicious, security-breaking program that is disguised as something benign" [5]. This paper considers a broader definition of a Trojan horse to include programs that are neither malicious nor security-breaking. Examples include "adware" that piggy-backs on more legitimate programs and anti-piracy programs that execute whenever certain applications are started. Thus, a Trojan horse is taken to mean any code on a system that is not explicitly allowed or expected to be run by the user.

A machine can become the host to a Trojan horse in several ways.

- A user may visit a website that exploits a browser or operating system flaw on the user's machine. An example is the Bofra vulnerability [14], which only requires a user to click on a link before executable content is downloaded to the user's machine.

- A use may download a program out of curiosity, leading to a compromise, even if the program is supposedly uninstalled later on.

- A user may open and execute an email attachment either unintentionally or because it comes from a trusted source. An example is the Sober worm [16] that uses social engineering to spread.

- A user may execute the preview, auto-accept or auto-get features of a program that lead to code execution and a Trojan horse being deposited on the user's system.

- An intrusion may leave Trojan horse replicas of important system utilities. This type of Trojan horse, called a rootkit, installs binaries that mask the compromise and leave a back-door into the system. Rootkits are freely available for a number of platforms.

- An untutored user might blindly follow instructions on a webpage that leave a Trojan horse on the system. Many users are deceived by phishing, which is the practice of using a site that appears legitimate to convince users to perform certain actions. For example, a "bank website" might request a user to download a "security update" for his computer.

Avoiding infection requires that users be certain about the source of a file and its contents, and that the system be secured and fully patched. Even these precautions may be insufficient if the user is fooled (e.g., by phishing) into compromising his own system. Alternatively, an attacker can take advantage of the time delay between an exploit being created and a patch being released.

2.2 Unauthorized Code

Unauthorized code is code that should not be running on a system. Examples include user-created binaries and games on a mail server. The difference between unauthorized code and a Trojan horse is that the former is installed on the system with the knowledge of the user while the latter is not.

Note that many privilege escalation attacks require that binaries be run by untrusted users with access to normal user privileges. An example is the privilege escalation attack that takes advantage of the `sudo` utility's failure to clean the environment [7]. This is an example of unauthorized code that is malicious and is also likely to be run by script kiddies on an otherwise secure system. Since there is almost inevitably a delay between the release of an exploit and the creation (and application) of a vendor-supplied patch or workaround, the intervening time leaves systems vulnerable to compromise. Of course, once a system is compromised, hiding the compromise using a rootkit is trivial.

2.3 Integrity Checking

Traditionally, guarding against Trojan horses and unauthorized code has been accomplished using a solution such as Tripwire [13], which takes "snapshots" of a system. These snapshots include relevant information about files that makes tampering easy to detect. After a suspected compromise, or simply as part of a daily security regimen, the integrity of files on the system is checked against the snapshot. Integrity checkers have varying levels of sophistication, with some (e.g., `mtree` [6]) checking file integrity as a side-effect of their main functionality.

Conventional integrity checkers suffer from the fact that a time lapse exists between the compromise and the check. This may be exploited by attackers who can launch further attacks using executables placed on the compromised machine. Therefore, integrity checkers are only a partial solution to the problems noted in the discussion of Trojan horses and unauthorized code (Sections 2.1 and 2.2).

It is also important to note that should a compromise occur, it may be difficult to verify the integrity of the database used for comparing

file characteristics. For this reason, the database should be stored either offline or on a different (and secure) system.

2.4 Pre-Execution Validation

Validating the integrity of binaries using digital signatures or simple hashes just before execution eliminates the time lapse problem suffered by integrity checkers. The following tools may be used to perform pre-execution validation.

2.4.1 CryptoMark. CryptoMark digitally hashes and signs an Executable and Linking Format (ELF) binary program, storing the result within a SHT_NOTE section [3]. It computes an MD5 hash of the loadable file segments, and checks the hash and signature via a kernel module whenever the executable is run. Userspace tools are used to perform and manage the signing of a file.

CryptoMark may be run in a number of configurations. One common configuration requires all binaries to be signed; unsigned binaries or those with incorrect signatures are not allowed to run. Another configuration requires all binaries that run as the superuser to be signed. This allows users to compile and run their own programs, but denies them the ability to compile and run binaries that run as the superuser, even if they have managed to gain superuser access.

2.4.2 WLF. WLF [4] verifies the integrity of ELF binaries in-kernel, whereas other techniques (e.g., [2]) verify binaries using modified interpreters. Key management is stressed by WLF, which makes provision for using signed binaries from different sources by embedding a `KeyID` field in the file signature. WLF can verify a large variety of files using a plug-in architecture.

2.4.3 TrojanXproof. TrojanXproof [15] verifies the integrity of ELF binaries using a secured database rather than signing the executables themselves. It takes the form of kernel patches for the FreeBSD and OpenBSD operating systems, and verifies shared libraries and ELF executables. TrojanXproof does not cryptographically secure files using digital signatures. Instead, it simply creates MD5 hashes of files and relies on the security of the database.

2.4.4 DigSig. DigSig [1] uses a kernel module to check the signatures generated by BSign [11], a tool for signing ELF binaries. All binaries must be signed correctly in order to run. DigSig caches signatures (also suggested in [2]), which makes the repeated use of commands

Table 1. Evaluation of existing solutions.

Name	Ease of Use	Executable Checking	Pre-Exec. Validation	Active Developmt.	Transprnt. Checking
Tripwire	X	X		X	
CryptoMark	X		X		X
WLF			X		X
TrojanXproof			X	X	X
DigSig	X		X	X	X
SELinux		X		X	X

much faster than it would be otherwise. In general, DigSig has most mature implementation of runtime validation.

2.4.5 Status of Tools. Of the tools described above, only DigSig appears to be under active development. CryptoMark, a project of Immunix [8], has disappeared from the vendor's website. Work on WLF has ceased, and the code that does exist is quite fragile. TrojanXproof is unsupported in most cases.

2.5 Comprehensive Security Solutions

Unauthorized binaries and Trojan horses can be defeated using comprehensive security solutions such as SELinux [9], which restrict executable access to known valid binaries and ensure fine-grained access control. Comprehensive security solutions, however, are complex to configure. For example, SELinux policies require an understanding of role-based access control, type-based enforcement, domains, access vectors and more. This makes them difficult for the average user to understand and use, resulting in an increased likelihood of misconfiguration.

2.6 Evaluation of Solutions

Table 1 provides an evaluation of existing solutions. "Ease of use" is determined by the amount of effort the solution takes to set up and maintain. "Transparent checking" refers to how transparent the checks are to the user. The other categories are self-explanatory.

3. Pre-Execution Validation Strategies

This section discusses strategies for pre-execution validation of files that can secure machines against Trojan horses and unauthorized code.

3.1 Basic Strategy

It is important that a system used to verify whether or not tampering has occurred is itself tamper-proof. In other words, it should not be possible for an attacker who has gained administrator privileges to subvert the system. The following two techniques are used for this purpose.

- **Digital Signatures:** Binaries are digitally signed using a private key that may be stored offline or protected with a password. Only properly-signed binaries are allowed to run. Without access to the private key, an attacker cannot sign his own binaries, preventing him from running unauthorized executables on the system.

- **In-Kernel Verification:** Signatures of binaries are verified in-kernel. A system kernel is one of the hardest components to compromise. It cannot (currently) be replaced without a reboot, so an attacker would have to reboot the machine – something that a system administrator is likely to notice. Also, the kernel image may be specific to the machine and, therefore, difficult to replicate.

  Creating a custom kernel that contains untrusted code is a non-trivial task. However, this situation may change with the introduction of system calls such as `kexec` in the Linux MM-series kernel that allows a running kernel to be replaced without rebooting the system. Such a "feature" should never find its way into a secure system. To address this issue, machines could boot off CDROMs or other secure read-only media that contain the kernel and base system, or they could use a kernel made available over a network. Alternatively, a kernel checksum could be placed on another machine, or the kernel could be cryptographically signed, and these could be checked during booting. These measures make compromising the kernel difficult, if not impossible.

Although we use the term "signing binaries" in this paper, note that that both binaries and libraries must be digitally signed. Signing ideally occurs by sending the binary to a trusted party who examines it byte-for-byte against a known good copy, and signs it if it matches. This process can be automated quite easily, using a web service or remote procedure call (RPC) variant that signs binaries.

To simplify the discussion, we consider the case of having one "correct" signature rather than a number of possible signatures. The techniques described below are easily adapted to handling more than one signature.

Signature revocation must be taken into account in all designs. Revocation occurs if an application is found to be exploitable or otherwise

unsuitable for use on a system. In the absence of revocation, an attacker who has saved an earlier signed version of the exploitable file could use it to replace the current version, opening up a security hole in the system. It should not be possible to remove signatures from a list of revoked signatures, nor should it be possible to add signatures to the list improperly. The former may lead to a compromise; the latter to denial of service as legitimate programs would be prevented from executing.

3.2 In-Binary Signatures

An ELF object file has a number of sections, some which are loaded into memory at runtime and some which are not. Sections that hold vendor-specific information are of type SHT_NOTE, and are not loaded into memory to form the executable image at runtime [12]. An ELF binary may have its signature stored in a specially-crafted SHT_NOTE section.

The advantage of this approach is that a binary and its signature are linked through one file; there is no need to have separate storage for the signature, which simplifies verification. Since SHT_NOTE sections are not loaded to form an executable image, the behavior of the executable is the same as if the section did not exist at all. This means that signed binaries are just as portable as unsigned binaries.

Signature revocation is problematic as there is no good way to keep track of a list of revoked signatures. The revoked signatures cannot be kept within the file as this would not get around the problem of file replacement. Also, the need to maintain a separate database that must be kept with the files removes the benefit of having a single storage site for signatures. In addition, the database grows as signatures are revoked; checking every signature against an ever-increasing "revoked" list does not scale well. Another drawback is that only ELF binaries may be checked; Python scripts, for example, do not have SHT_NOTE sections.

3.3 External Signatures

Keeping signatures separate from executable files means that two files – the binary itself and the signature database – must be opened whenever a binary (executable or library) is run. The database is secured by signing it with a private key; the file hashes within the database need not be encrypted individually. Signing is implemented by sending the executable and the database to a third party. The third party verifies the file, unlocks the database, adds the new signature or replaces the old one, and returns only the database.

One advantage of this approach is that any executable, not just ELF files, can be checked. Meta-information about the file (e.g., user/group ownership) can be stored and checked along with the signature for added security. Revocation is not an issue as replacing a file with a previous version is the same as having a file with an invalid signature; therefore, no special revocation check needs to be done. To ensure that the database is not replaced by an earlier version, it is necessary to maintain a version number in-kernel and in the database, or to have the database located on secure read-only media (e.g., locked flash memory). The database itself can be secured in a number of ways, as it is only one file rather than hundreds of binaries.

A disadvantage of this approach is that opening and checking two files upon every execution is slower than simply dealing with one file.

3.4 Caching

Validating an executable file before execution leads to an inevitable delay at program startup. While the delay may not be noticeable for a large executable, e.g., a word-processing program or a web browser, it can be quite significant for small system utilities such as `ls` or `grep`. The problem can be ameliorated by caching signatures, especially in the case of external signatures.

A signature cache should be invalidated after writing to a file, removing a file from a directory hierarchy, and rebooting. In addition, networked files to which writes cannot be reliably detected should never have their signatures cached. This approach to caching is taken by DigSig, which has contributed to significant increases in speed [1]. In contrast, CryptoMark does no caching [3]; it was removed from public distribution in 2004 due to sluggish performance.

3.5 Limitations

Performance limitations are significant in the case of networked file systems. As indicated in Section 3.4, it is difficult to maintain good performance without caching. Cached signatures are valid only when a kernel can record accesses to a file system. Since this is not possible for networked file systems, signature caching cannot be employed.

This approach is intended to increase system security; it is not a comprehensive security solution, e.g., SELinux. Nevertheless, the approach has several advantages, including transparency, ease of use and minimal setup time, all of which contribute to ease of deployment across machines intended for very different purposes. However, the trade-off is

that customization and the benefits of a comprehensive, flexible security policy are lost.

Of course, all security is lost if the kernel or the third party who holds the private key are compromised. As discussed in Section 3.1, several techniques exist for securing the kernel. Securing the third party is outside the scope of this work.

4. Digital Forensics

Of course, denying an executable permission to run is simply one response to an invalid or nonexistent signature. Several other options are possible – from logging the behavior of the executable to notifying the system administrator. The proposed technique can be used in a honeypot implementation to log executable files that do not have signatures, make backup copies of them and maintain records of their behavior. It can also be used in conjunction with a network traffic logger to record the behavior of certain exploits "in the wild" for later analysis, for input to an intrusion detection system (IDS) or as evidence of malicious activity.

In the case of a suspected intrusion, system tools can be limited to a known good set by disallowing the execution of all unsigned binaries. In this state the system can be checked for a compromise. While other methods exist for locking down a system, the benefits of this approach are that it allows read-write access to the system and ensures the integrity of the tools used for analysis. The system can be locked down as soon as suspicious behavior is noted by an intrusion detection system. If a production web server is compromised at night or on a weekend, the lockdown strategy ensures that the server continues to operate while the threat of further compromise is mitigated and unsigned executables are stored in a safe location for further analysis.

5. Conclusions

The focus of this work is protecting systems from Trojan horses and unauthorized code. However, the approach has several other applications, including assisting with honeypot implementations, incident response and forensic investigations. Other variations, such as only allowing signed binaries to be executed in a superuser context, are also possible.

The approach is intended to increase system security; it is not a comprehensive security solution, e.g., SELinux. Nevertheless, the approach has several advantages, including simplicity, transparency, ease of use and minimal setup time.

References

[1] A. Apvrille, D. Gordon, S. Hallyn, M. Pourzandi and V. Roy, The DigSig Project, *LinuxWorld Magazine*, vol 2(1), December 22, 2003.

[2] W. Arbaugh, G. Ballintijn and L. van Doorn, Signed executables for Linux, Technical Report CS-TR-4259, University of Maryland, College Park, Maryland, 2001.

[3] S. Beattie, A. Black, C. Cowan, C. Pu and L. P. Yang, CryptoMark: Locking the stable door ahead of the Trojan horse, Technical Report, WireX Communications Inc., Portland, Oregon, 2000.

[4] L. Catuogno and I. Visconti, A format-independent architecture for run-time integrity checking of executable code, in *Security in Communication Networks, Lecture Notes in Computer Science, Volume 2576*, S. Cimato, C. Galdi and G. Persiano (Eds.), Springer, Berlin-Heidelberg, pp. 219-233, 2003.

[5] FOLDOC, Trojan horse, *FOLDOC: The Free On-Line Dictionary of Computing* (www.foldoc.org/foldoc/foldoc.cgi?query=Trojan+Horse&action=Search).

[6] FreeBSD, `mtree(8)`, *FreeBSD 5.3 System Manager's Manual*, January 11, 2004.

[7] L. Helmer, Sudo environment cleaning privilege escalation vulnerability (secunia.com/advisories/13199).

[8] Immunix Inc. (www.immunix.org).

[9] National Security Agency, Security-Enhanced Linux (www.nsa.gov/selinux).

[10] B. Paul, *Evaluation of Security Risks Associated with Networked Information Systems*, Master's Thesis, School of Business Administration, Royal Melbourne Institute of Technology, Melbourne, Australia, 2001.

[11] M. Singer, `bsign(1)`, The Debian Project (packages.debian.org/testing/admin/bsign), 2001.

[12] Tool Interface Standards Committee, Executable and Linkable Format (ELF), Technical Report, Unix System Laboratories, Summit, New Jersey, 2001.

[13] Tripwire Inc., Tripwire for servers datasheet, Technical Report, Tripwire, Inc., Portland, Oregon (www.tripwire.com/files/literature/product_info/Tripwire_for_Servers.pdf), 2005.

[14] B. Wever and ned, Microsoft Internet Explorer malformed IFRAME remote buffer overflow vulnerability (securityresponse.symantec.com/avcenter/security/Content/11515.html).

[15] M. Williams, Anti-Trojan and Trojan detection with in-kernel digital signature testing of executables, Technical Report, NetXSecure NZ Limited, Canterbury, New Zealand, 2002.

[16] C. Wueest, W32.Sober.I@mm (sarc.com/avcenter/venc/data/w32.sober.i@mm.html).