IFIP AICT 337

Kam-Pui Chow
Sujeet Shenoi
(Eds.)

Advances in Digital Forensics VI

Springer

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.

Kam-Pui Chow   Sujeet Shenoi (Eds.)

Advances in Digital Forensics VI

Sixth IFIP WG 11.9 International Conference
on Digital Forensics
Hong Kong, China, January 4-6, 2010
Revised Selected Papers

Springer

Volume Editors

Kam-Pui Chow
University of Hong Kong, Department of Computer Science
Hong Kong, China
E-mail: chow@cs.hku.hk

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

Library of Congress Control Number: 2010934317

CR Subject Classification (1998): H.3, C.2, K.6.5, D.4.6, F.2, E.3

ISSN 1868-4238
ISBN-10 3-642-15505-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-15505-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 219/3180

Contents

Contributing Authors

Saif Al-Kuwari is a Ph.D. student in Mathematics with the Information Security Group at Royal Holloway, University of London, London, United Kingdom. His research interests are in the area of digital forensics, particularly clandestine localization and tracking in MANET and VANET environments.

Ivan Burke is an M.Sc. student in Computer Science at the University of Pretoria, Pretoria, South Africa; and a Researcher with the Council for Scientific and Industrial Research, Pretoria, South Africa. His research interests include wireless networks and agent-based modeling.

Hilton Chan is an Adjunct Assistant Professor of Information Systems, Business Statistics and Operations Management at the Hong Kong University of Science and Technology, Hong Kong, China. His research interests include cyber crime investigations, digital forensics, incident response and crisis management.

Patrick Chan is an M.Phil. student in Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include web security, applied cryptography and network security.

Jenny Chen is a Research Engineer with the Center for Information Security and Cryptography at the University of Hong Kong, Hong Kong, China. Her research interests include digital forensics and peer-to-peer networks.

Lijuan Chen is an Associate Researcher at the Shandong Computer Science Center, Jinan, China. Her research interests include information security, digital forensics and database systems.

Zhenya Chen is an Associate Researcher at the Shandong Computer Science Center, Jinan, China. Her research interests include digital forensics and grid computing.

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

Fred Cohen is the Chief Executive Officer of Fred Cohen and Associates; and the President of California Sciences Institute, Livermore, California. His research interests include digital forensics, information assurance and critical infrastructure protection.

Michael Cohen is a Data Specialist with the Australian Federal Police in Brisbane, Australia. His research interests include network forensics, memory forensic analysis, large-scale forensic frameworks and the AFF4 forensic file format.

Scott Conrad is a Senior Digital Forensics Research Assistant at the National Center for Forensic Science, University of Central Florida, Orlando, Florida. His research interests include personal gaming/entertainment devices and virtualization technologies.

Philip Craiger is an Associate Professor of Engineering Technology at Daytona State College, Daytona Beach, Florida; and the Assistant Director for Digital Evidence at the National Center for Forensic Science, University of Central Florida, Orlando, Florida. His research interests include the technical and behavioral aspects of information security and digital forensics.

Greg Dorn is a Senior Digital Forensics Research Assistant at the National Center for Forensic Science, University of Central Florida, Orlando, Florida. His research interests include virtualization technologies and personal gaming/entertainment devices.

Isao Echizen is an Associate Professor of Computer Science at the National Institute of Informatics, Tokyo, Japan. His research interests include media security, information processing and information hiding.

Ryo Furukawa is a Researcher at NEC Corporation, Tokyo, Japan. His research interests include access control, privacy management and optimization.

Pavel Gladyshev is a Lecturer of Computer Science and Informatics at University College Dublin, Dublin, Ireland. His research interests include information security and digital forensics.

Paolo Gubian is an Associate Professor of Electrical Engineering at the University of Brescia, Brescia, Italy. His research areas include integrated circuit design, digital forensics and embedded systems security.

Murat Gunestas is a Police Major with the General Directorate of Security in Ankara, Turkey. His research interests include web services security, computer and network forensics, and software engineering.

Yinghua Guo is a Postdoctoral Research Fellow at the School of Computer and Information Science, University of South Australia, Adelaide, Australia. His research interests include digital forensics, information assurance, network security, intrusion detection systems and wireless networking.

Wing-Kai Hon is an Assistant Professor of Computer Science at National Tsing Hua University, Hsinchu, Taiwan. His research interests include data compression, design and analysis of algorithms, and combinatorial optimization.

Lucas Hui is an Associate Professor of Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include computer security, cryptography and digital forensics.

Ricci Ieong is a Ph.D. student in Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include digital forensics, peer-to-peer forensics and time correlation analysis.

Joshua James is a Ph.D. student in Computer Science and Informatics at University College Dublin, Dublin, Ireland. His research interests include cyber crime investigation process models and standards, evidence correlation techniques, human inference and event reconstruction.

Masahiro Kawato is an Assistant Manager at NEC Corporation, Tokyo, Japan. His research interests include distributed computing, enterprise systems security and privacy management.

Michael Kwan is a Ph.D. student in Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include digital forensics, digital evidence evaluation and the application of probabilistic models in digital forensics.

Pierre Lai is a Ph.D. student in Computer Science at the University of Hong Kong, Hong Kong, China. Her research interests include cryptography, peer-to-peer networks and digital forensics.

Frank Law is a Ph.D. student in Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include digital forensics and time analysis.

Frankie Li is an M.Sc. student in Electronic Commerce and Internet Computing at the University of Hong Kong, Hong Kong, China. His research interests include digital forensics and malware analysis.

Fumio Machida is an Assistant Manager at NEC Corporation, Tokyo, Japan. His research interests include dependable computing, autonomic computing and systems management.

Yoshiharu Maeno is a Principal Researcher at NEC Corporation, Tokyo, Japan. His research interests include the analysis and control of complex distributed systems for communications and computing.

Murad Mehmet is a Ph.D. student in Information Technology at George Mason University, Fairfax, Virginia. His research interests include network security, digital forensics and web services security.

Yuki Nakayama is an M.S. student in Computer Science at Keio University, Kanagawa, Japan. His research interests include network security and digital forensics.

Sipho Ngobeni is an M.Sc. student in Computer Science at the University of Pretoria, Pretoria, South Africa; and a Researcher with the Council for Scientific and Industrial Research, Pretoria, South Africa. His research interests include network security, digital forensics and information security.

Kenichi Okada is a Professor of Information and Computer Science at Keio University, Kanagawa, Japan. His research interests include computer-supported cooperative work, groupware, human-computer interaction and ubiquitous computing.

James Okolica is a Ph.D. student in Computer Science at the Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio. His research interests include text mining and memory forensics.

Richard Overill is a Senior Lecturer in Computer Science at King's College London, London, United Kingdom. His research interests include digital forensics, cyber crime analysis, anomaly detection, cyber warfare and information security management.

Gilbert Peterson is an Associate Professor of Computer Science at the Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio. His research interests include digital forensics, steganography and machine learning.

Mark Pollitt, Chair, IFIP Working Group 11.9 on Digital Forensics, is an Associate Professor of Engineering Technology at Daytona State College, Daytona Beach, Florida; and a principal with the National Center for Forensic Science, University of Central Florida, Orlando, Florida. His research interests include forensic processes, knowledge management, information security and forensic quality management.

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

Antonio Savoldi is an Associate Researcher in the Department of Information Engineering at the University of Brescia, Brescia, Italy. His research interests include digital forensics, embedded systems security and counter-forensic methodologies.

Bradley Schatz is the Director of Schatz Forensic, a digital forensics consultancy; and an Adjunct Associate Professor at the Information Security Institute, Queensland University of Technology, Brisbane, Australia. His research interests include volatile memory acquisition, scaling storage forensics and the development of the new AFF4 forensic evidence container.

Seiji Shibaguchi is a Software Developer with Nintendo in Kyoto, Japan. His research interests include network security and digital forensics.

Jantje Silomon is a Research Associate in the Department of Computer Science, King's College London, London, United Kingdom. Her research interests are in the area of digital forensics.

Jill Slay is the Dean of Research and a Professor of Forensic Computing at the University of South Australia, Adelaide, Australia. Her research interests include information assurance, digital forensics, critical infrastructure protection and complex system modeling.

Kumiko Tadano is a Researcher at NEC Corporation, Tokyo, Japan. Her research interests include dependable computing, systems management and enterprise systems security.

Benjamin Tang is an undergraduate student in Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include digital forensics.

Vrizlynn Thing leads the Digital Forensics Research Group at the Institute for Infocomm Research, Singapore. Her research interests include digital forensics, network security, intrusion detection and mitigation, networking protocols and optical fiber communications.

Hayson Tse is a Ph.D. student in Computer Science at the University of Hong Kong, Hong Kong, China. His research interests are in the area of digital forensics.

Hein Venter is an Associate Professor of Computer Science at the University of Pretoria, Pretoria, South Africa. His research interests include network security, digital forensics and information privacy.

Chunxiao Wang is an Associate Researcher at the Shandong Computer Science Center, Jinan, China. Her research interests include cryptography, digital forensics and software engineering.

Kenny Wang is a Ph.D. candidate in Law at the University of Hong Kong, Hong Kong, China. His research interests include digital evidence, cyber crime and legal jurisdiction.

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

Stephen Wolthusen is a Professor of Information Security at the Norwegian Information Security Laboratory, Gjovik University College, Gjovik, Norway; and a Reader in Mathematics at Royal Holloway, University of London, London, United Kingdom. His research interests include the modeling and simulation of critical infrastructures, and network and distributed systems security.

Ying Yang is an Associate Professor of Information Security at the Shandong Computer Science Center, Jinan, China. Her research interests include computer security, digital forensics and software systems.

Siu-Ming Yiu is an Assistant Professor of Computer Science at the University of Hong Kong, Hong Kong, China. His research interests include bioinformatics, computer security, cryptography and digital forensics.

Yuandong Zhu is a Ph.D. student in Computer Science and Informatics at University College Dublin, Dublin, Ireland. His research interests include user activity analysis and forensic tool development.

Preface

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

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

This volume contains twenty-one edited papers from the Sixth IFIP WG 11.9 International Conference on Digital Forensics, held at the University of Hong Kong, Hong Kong, January 4–6, 2010. The papers were refereed by members of IFIP Working Group 11.9 and other internationally-recognized experts in digital forensics.

The chapters are organized into six sections: themes and issues, forensic techniques, Internet crime investigations, live forensics, advanced forensic techniques and forensic tools. The coverage of topics highlights the richness and vitality of the discipline, and offers promising avenues for future research in digital forensics.

This book is the result of the combined efforts of several individuals. In particular, we thank Daniel Guernsey, Pierre Lai and Catherine Chan for their tireless work on behalf of IFIP Working Group 11.9. We also acknowledge the support provided by the University of Hong Kong, Hong

Kong Police Force, Hong Kong Forensic Science Foundation, National Security Agency, Immigration and Customs Enforcement, and U.S. Secret Service.

KAM-PUI CHOW AND SUJEET SHENOI

I

THEMES AND ISSUES

Chapter 1

A HISTORY OF DIGITAL FORENSICS

Mark Pollitt

Abstract The field of digital forensics is relatively new. While its history may be chronologically short, it is complex. This paper outlines the early history of digital forensics from the perspective of an early participant. The history is divided into four epochs: pre-history, infancy, childhood and adolescence. Each of these epochs is examined from the perspective of the people involved, the criminal targets, the forensic tools utilized, the organizational structures that supported digital forensic practitioners and how the community formed. This history is, by necessity, incomplete and biased. There is a need for rigorous historical research in this area before all traces of the past are forgotten or obliterated.

Keywords: Digital forensics, history

1. Introduction

One of the important rules in public speaking is to never begin with an apology. I have often broken this rule when delivering speeches and will do so again in this paper. My audience for this paper will be – for the most part – scientists, who expect well-documented foundations, reliable data and rigorous logic. Hence, my apology: there is little, if any, of these things in this paper.

It would be tempting to frame this exploration as historical research. And while I have done some research, which is duly referenced, this is fundamentally a personal history, which I have lived since the early 1980s. I am biased and less than fully informed; I have an unreliable memory and selective recall. Those of us who worked in this field from the beginning gave little thought to documenting a history. We were just trying to do our jobs.

So why should anyone read this paper? There are several reasons. For those who currently work in digital forensics and the students who are just entering the field, it is important to understand how we got here. During my tenure as a criminal investigator, one of my wisest informants insisted on teaching me the history of Baltimore, its ways and its people. He said, very correctly, that if I did not understand how people got where they did, I would not understand what they were doing now, what they might do in the future and why.

To those of us who have an interest in history, especially related to cyber crime and digital forensics, this paper is merely a starting point with many gaps that have to be filled. In some small measure, I hope this paper will serve as motivation to correct and expand the history as well as begin a dialog about the field, its past and its future. We must not delay – digital forensics is almost three decades old and many of the original players are moving on.

2. Epochs and Lenses

Even the simplest history has to have at least three phases: before, during and after. It may be called something more prosaic, but there is an inherent need for logical structure, even in something as illogical as history. For this story, I have chosen to use the notion of "epochs." It is, of course, entirely arbitrary on my part but it has both factual and logical bases. The epochs are: pre-history, infancy, childhood and adolescence.

Like traditional history, it is useful to explore a time period using a particular perspective. These are often called "lenses," as a metaphorical attempt to focus both the writer and the reader on specific elements of the historical data. In reviewing the history of digital forensics, I realized that there were some critical elements that combined to create the discipline. In my view these are: people, targets, tools, organizations and the community as a whole. I make no assertions that they constitute the totality of the history, but they are key vectors that help capture the essential elements of the history.

Modern electronic computers evolved during the second half of the 1900s. The History of Computing Project defines 1947 as the beginning of the Industrial Era of Computing [18] and we are still in the midst of this era. So much has happened in computing since 1947 that it is helpful to break it down into manageable chunks. In particular, digital forensics – or forensic computing as some like to call it – has a shorter history. As a result, I choose to make some further arbitrary divisions based on events that were significant to the digital forensic community.

3. Pre-History

The first epoch, which I label as pre-history, covers the period prior to 1985. It is not surprising that this is the least documented epoch because much of what happened was not focused on digital forensics. In fact, the term simply did not exist. From the 1960s until the early 1980s, computers were primarily an industrial appliance, owned and operated by corporations, universities, research centers and government agencies. They required a large physical infrastructure, including massive amounts of power and air conditioning, and highly skilled, dedicated staff. Their function was largely data processing. It is in this role that computers first became of interest to the information security, legal and law enforcement communities.

Donn Parker's 1976 book, *Crime by Computer*, is perhaps the first description of the use of digital information to investigate and prosecute crimes committed with the assistance of a computer. System administrators were, for the most part, responsible for the security of their own systems, most of which were not significantly networked to the outside world. System audits were designed to ensure the efficiency and accuracy of data processing, which was very expensive at the time. In effect, these audits constituted the first systematic approach to computer security. A byproduct of these efforts was that information collected during audits could be used to investigate wrongdoing [12]. This was not totally lost on the law enforcement community.

Organizations such as the Department of Defense, Internal Revenue Service (IRS) and Federal Bureau of Investigation (FBI) created *ad hoc* groups of volunteer law enforcement agents, who were provided with rudimentary mainframe and mini-computer training. These computer-trained investigators would assist other case investigators in obtaining information (primarily) from mainframe computers – stored data and access logs. Usually, the computer-trained investigators would work in cooperation with systems administrators.

Cliff Stoll's 1990 book, *The Cuckoo's Egg* [16], captures the practice and the ethos of early digital forensics. It also highlights the reluctance of government agencies to engage in this new area. It was difficult for traditional managers and investigators to grasp the potential of computers to be both tools and victims of crime. Stoll, then a Unix systems administrator, was attempting to reconcile two system accounting programs that were reporting a small difference in usage. After much investigation, he realized that hackers were accessing a large number of computers, including some sensitive systems. Using system administration tools and considerable experimentation, he developed, on his own initiative, a method for recording the hackers' malicious activities in real time.

Ad hoc and individual are the defining characteristics of the first epoch. There were virtually no dedicated organizations, procedures, training or tools specifically designed for digital forensics. Operating system tools and utilities were utilized along with traditional scientific and investigative problem-solving approaches.

4. Infancy (1985-1995)

The advent of the IBM PC in the early 1980s resulted in an explosion of computer hobbyists. The PCs, while powerful, had relatively few applications and were not, despite the advertising copy, user-friendly. Many of these hobbyists had previously worked with Commodore 64s, Radio Shack TRS-80s and Ataris. These early computers enabled hobbyists to write program code and access the internals of the operating systems and hardware. These skills were channeled to the new IBM PCs and PC-compatible computers.

Among the hobbyists were law enforcement personnel from a wide variety of organizations. Some of the key individuals were Mike Anderson, Danny Mares and Andy Fried from the IRS; Ron Peters and Jack Lewis from the U.S. Secret Service; Jim Christy and Karen Matthews from the Department of Defense; Tom Seipert, Roland Lascola and Sandy Mapstone from local U.S. law enforcement agencies; and the Canadians, Gord Hama and Steve Choy. Many of them became charter members of the first organization (to my knowledge) dedicated to digital forensics – the International Association of Computer Investigative Specialists (IACIS).

There were many other individuals as well. What they all shared was an understanding that computers would play a critical role in criminal investigations and, specifically, that computers are important sources of evidence. All these individuals believed this to the extent that they spent much of their own time and money to learn about new computing technologies. Their agencies were not supportive of their efforts, but we owe these individuals a debt of gratitude for the personal and financial investments that they made. Without their inspired and timely efforts, much of what we do today in the discipline of digital forensics would not be possible.

The early efforts were by no means limited to North America. Law enforcement officials in Europe, Asia and Oceania were struggling with the same problems and making the same personal commitment to prepare themselves and their organizations for the future that they knew was coming. In 1993, the FBI hosted the First International Conference on Computer Evidence at the FBI Academy in Quantico, Virginia, which was attended by representatives from 26 countries. At this conference, it was agreed that the community needed to band together at the agency level to coordinate efforts, share experience and provide assistance to each other. In 1995, the second conference was held in Baltimore and the International Organization on Computer Evidence (IOCE) was founded [21].

The cases investigated by these pioneers were very basic by today's standards. Much of the focus was on recovering data from standalone computers. Data recovery was a major issue because storage was costly and users routinely deleted data and re-formatted media. The Internet was not yet popular, but criminals were using dial-up access to compromise computers.

The use of inexpensive computers to hack the telephone system was a new dimension of fraud. Telephone service was billed by distance and use. Criminals and adolescents found that by hacking telephone networks they could obtain "free" telephone service as well as previously unavailable levels of anonymity [6].

The subjects of computer crime investigations were generally traditional criminals who used computers to support their activities or young people who used their technical skills to illegally obtain computer access and software. While IBM PCs and PC-compatible computers running DOS and early Windows variants were the most commonly encountered devices, early Apple products as well as Commodore and Atari computers were often encountered.

The tools used by the pioneering investigators included home-grown, command line tools and commercial products adapted to forensic use. Andy Fried's IRS Utilities, Steve Mare's Maresware, Steve Choy's IACIS Utilities and Gord Hama's RCMP Utilities were all command line tools that were distributed within the law enforcement community. Each of these tools tended to solve a specific digital forensic problem, such as imaging or identifying deleted files. Some of the later variants, like Hama's REDX, allowed for multiple operations and rudimentary piping. Norton Utilities and PC Tools, both commercial products designed for data recovery and file management, proved to be very powerful tools for digital forensics and virtually all the forensic training during the epoch utilized one or both of these tools. Another noteworthy product of this period was SafeBack, which was created by Chuck Guzis in 1991 to acquire forensic images of evidence. SafeBack may well have been the first commercial digital forensic product.

During this epoch, digital forensic practitioners conducted their examinations wherever they could find space. Often it was at their desks, in their basements at home or in unused storage space. The notion of a purpose-built laboratory was years away. Even large law enforcement agencies had hardly any funds for equipment – examiners had to use surplus equipment or the very equipment that they seized.

At the time, digital forensics was an arcane area that operated in direct conflict with the geographically- and statutorily-bound practice of criminal investigations. Criminals operated across city, state and national boundaries in almost real time, but investigators had no choice but to communicate and work directly with their peers, wherever they were located.

Digital forensic practice also operated in direct conflict with the traditional, laboratory-based practice of forensic science. However, some agencies did see the need for digital forensic capability. The IRS created the Seized Computer Evidence Recovery Specialist (SCERS) Program, the U.S. Secret Service its Electronic Crimes Special Agent Program (ECSAP), the FBI its Computer Analysis Response Team (CART), and the U.S. Air Force Office of Special Investigations its Computer Crime Investigator (CCI) Program and what eventually became the Defense Computer Forensic Laboratory (DCFL). Each agency adopted a different model of selection, training and operations based on its structure and culture. But these agencies were the exception; the majority of digital forensic investigations were performed by individual officers with minimal training, often using personal equipment, and without any supervision or formal quality control.

But the digital forensic community was growing. In addition to IACIS, many grassroots efforts were underway to pool knowledge, resources and talent. In the Midwestern United States, the Forensic Association of Computer Technologists (FACT) created training opportunities and a network of geographically-dispersed practitioners. In the Baltimore area, forensic practitioners from the FBI, U.S. Secret Service, Maryland State Police and Baltimore County Police started an *ad hoc* organization called "Geeks with Guns." In the United Kingdom, practitioners from many law enforcement agencies formed the Forensic Computing Group (FCG) under the auspices of the Association of Chief Police Officers (ACPO). It was during this epoch that the High Tech Crime Investigation Association was formed.

Forensic training was developed and offered by these organizations as well as by some of the larger law enforcement agencies. The demand for quality, affordable training far exceeded the availability, a situation that continued to plague the field of digital forensics for many years (some would argue that it continues to this day). During this period, the academic community was almost totally disinterested in the field, with two notable exceptions: Gene Spafford, from Purdue University and Dorothy Denning, then at Georgetown University. These two professors encouraged many law enforcement agents and students to venture into this important new field.

5. Childhood (1995-2005)

The next decade proved to be one of tremendous growth in size and maturity. This growth had numerous drivers, but there were three that had the most significance.

The first driver was the explosion of technology that occurred during the epoch. Computers became ubiquitous, cell phones became essential and the Internet became the world's central nervous system. At the beginning of the epoch, most voice calls were via landline, most computer network connections were via dial-up and most people had not heard of the Internet. By the end of the epoch, almost everyone had an email address, a cell phone, relied on the Internet, and most homes and businesses had networks. Computer technology was embedded in virtually every element of daily life and that included criminal activities.

The second driver was the explosion of child pornography cases. This can be traced back to the George Stanley Burdynski, Jr. case in 1993. The investigation revealed that computers were used to traffic in illegal images of minors and led to the establishment of an online undercover operation called Innocent Images in 1995. Ten years later, there was a separate child pornography task force in half of all FBI offices; many other law enforcement agencies also operated their own task forces. This "new" violation resulted in the seizure of ever-increasing volumes of digital evidence and was a major driver in the growth of digital forensics [4, 17].

The highly anticipated Y2K problem proved to be a non event from a computer perspective, but the events of September 11, 2001 rocked the digital forensic world, just as it did the world at large. While computers played little direct role in the hijackings, investigators would find bits and pieces of evidence on computers around the world. The terrorists were using computers in the same ubiquitous ways as everyone else. This was further reinforced on the battlefields of Iraq and Afghanistan. The intelligence community, law enforcement and the military realized that the lack of digital forensic capabilities was a blind spot that needed immediate attention. The amount of time, money, people and resources devoted to digital forensics increased to massive levels.

At the beginning of this epoch, digital forensic practitioners were typically self-declared professionals or "resident experts," a term used to describe individuals who were seemingly effective computer users. With increasing volume, technical sophistication and legal scrutiny, it became increasingly important to carefully select and train digital forensic practitioners. The field itself began to become even more specialized. Digital audio, video and embedded devices such as cell phones required specific knowledge and training, separate from traditional storage media and network-focused forensics. Even these two fields were beginning to diverge at some levels, as the study of network intrusions became ever more complex. The discipline of digital forensics began to be driven by government agencies and professional organizations rather than by individuals.

The formalization of digital forensics made great strides during this epoch. The IOCE, G-8 High Tech Crime Subcommittee and Scientific Working Group on Digital Evidence (SWGDE) all published digital forensic principles between 1999 and 2000 [3, 7, 10, 15]. Going beyond mere principles, the American Society of Crime Laboratory Directors – Laboratory Accreditation Board (ASCLD-LAB), in cooperation with the SWGDE, recognized digital evidence as a laboratory discipline. In 2004, the FBI's North Texas Regional Computer Forensic Laboratory became the first ASCLD-LAB accredited digital forensic laboratory [1, 11].

Meanwhile, forensic tools underwent a metamorphosis. The homegrown, command line tools of the earlier epoch became complex, graphical user interface suites. The first of the new tools was Expert Witness, a product designed by Andy Rosen for Macintosh forensics that evolved into EnCase. EnCase, along with Forensic ToolKit (FTK), became commercial successes and are now standard forensic tools.

Several U.S. Government agencies also took on the task of developing tools. The FBI's Automated Case Examination System (ACES) and IRS's iLook tool had some success. However, the ability of commercial entities to evolve their products in step with advancing technology doomed the agency-developed tools to obsolescence. Meanwhile, the open source community stepped up, developing Linux tools such as Helix, Sleuth Kit and Autopsy Browser.

The digital forensic community likewise underwent a maturation process. Forensic services were being provided by a wide variety of entities, organized in a wide array of structures. Traditional forensic laboratories began offering digital examinations. The Department of Defense created its central Defense Computer Forensic Laboratory (DCFL) to service the law enforcement, intelligence and operational needs of the U.S. military [19]. The FBI started building a constellation of joint (federal, state and local law enforcement) laboratories dedicated to digital forensics – the Regional Computer Forensic Laboratories (RCFLs) [13]. Each laboratory would provide service to a geographic area and operate according to ASCLD-LAB standards. The U.S. Secret Service established a network of Electronic Crimes Task Forces, modeled on the highly-effective New York entity [20]. These task forces would provide investigative and forensic services within their area of operations. Many law enforcement agencies also developed dedicated units to handle digital forensic investigations.

6. Adolescence (2005-2010)

Since 2005, digital forensics has grown in depth and breadth. It has far more practitioners, performing many more examinations of a wider variety, involving ever larger amounts of evidence. In 2006, the United States Courts adopted new Rules for Civil Procedure that defined digital information as a new form of evidence and implemented a mandatory system, called electronic discovery or "eDiscovery," for dealing with digital evidence [5]. The workload in traditional law enforcement mushroomed. In Congressional testimony, the FBI announced that its Computer Analysis and Response Team (CART) examined more than 2.5 petabytes of evidence in 2007 alone [9].

Information security professionals now recognize digital forensics as a core skill area. While their objectives and needs often differ from those of law enforcement, the concepts and tools are often identical.

The digital evidence practitioners of today are far more likely to have had academic preparation in addition to formal training. They are likely to hold an array of certifications. Digital forensics is now considered "career enhancing," a far cry from just a few years ago.

Academic programs continue to spring up across the globe. While research funding for digital forensics lags other more traditional disciplines such as information security, colleges and universities have recognized the popularity and marketability of digital forensic education. Recently, the Forensic Education Program Accreditation Commission (FEPAC) took the first steps toward accrediting U.S. academic programs in digital forensics. The American Society of Testing Materials (ASTM) Technical Committee E-30 formulated a draft standard for digital forensic education and training programs [2].

Another measure of the academic health of a field is the number and quality of conferences. The Digital Forensic Research Workshop (DFRWS) is in its tenth year. The International Federation for Information Processing (IFIP) Working Group 11.9 on Digital Forensics is in its seventh year and the International Conference on Systematic Approaches to Digital Forensic Engineering just celebrated its fifth anniversary.

The materials that are being examined have also matured. Virtually every device that uses electricity now has some form of digital storage. Wired or wireless networks connect many of the devices that we use in our daily lives. This, in turn, has driven the development of many network- and web-based services, including cloud computing. Some of these services, such as Facebook and Twitter, are changing the way in which people interact. This change is starting to drive how digital evidence is collected. Email has already become a major source of probative information and a forensic and information management challenge [8].

During this epoch, forensic suites, most notably EnCase and FTK, have moved into the network/enterprise environment. They are being deployed in a prospective fashion in enterprises for corporate security and electronic discovery purposes. These same tools are being adapted to work in the emerging forensic environment of virtualized laboratories (using products such as VMWare) and storage area networks (SANs). Meanwhile, high-speed networks are being utilized to support the online review of evidence by investigators and prosecutors. The market for electronic discovery is booming, with vendors developing proprietary tools to automate the extraction and review of digital information.

This is an interesting time for the digital forensic community as a whole. The law enforcement, military and intelligence communities have designed organizational structures and processes to support their mission view. While there are some in these communities that are considering the impact of future technologies, there is far less emphasis on how targets will utilize these technologies and how customers will utilize the forensic products. Defining the forensic products of the future is another challenge. A strong tactical approach exists, but a long-term strategic plan is missing.

7. The Future

Predicting the future is a fool's game. Knowing that I will be wrong about many of my predictions, I will play the fool.

Digital forensics is a complex and evolving field. The practitioners of the future will be even better educated and trained; they will be team players trained to perform specific aspects of the forensic process. Forensics will no longer be a linear process focused on recovering data, but an evidence-based knowledge management process that will be integrated into investigations, intelligence analysis, information security and electronic discovery. There will be career digital forensic educators and researchers in addition to practitioners and managers. I fear that this new generation will be unaware of the early history and pioneering spirit that propelled the early years.

Our adversaries will be better organized, funded and educated. Criminals have recognized the value of distributed and collective efforts. Their payoffs will be significantly larger as the value of access and information grows along with society's dependence on the information infrastructure. Everyone (and their information) everywhere will be at risk at all times. Safety and security will be constantly threatened.

To counter these threats, digital forensic tools will have to improve. To overcome the sheer volume, the tools will have to be automated. In addition to performing data recovery, the tools will need to have built-in analytical capabilities, enabling important items to be identified without having to view every item. The tools will have to be semiotic, understanding human language and communications, and able to interpret content and context.

The organizations that employ digital forensic practitioners and those who rely on them will have to evolve as well. They will need to be accredited, with strong quality management and individual certifications. Much will ride on their reported results. Society needs assurance that the information collected and the conclusions reached are reliable. Organizations will have to cooperate and support the interoperability of people, tools and processes. Given the global scope of the problem, international legal standards will have to evolve.

8. Conclusions

In less than thirty years, digital forensics has blossomed from the germ of an idea, nurtured by brave pioneers, developed and expanded by professionals, to its current state. Many individuals have contributed their efforts, knowledge and enthusiasm to give the discipline a solid foundation for the future.

I apologize to the many worthy people who I have failed to mention in this narrative out of ignorance or forgetfulness. I earnestly hope that others will correct my errors, fill in the gaps and extend this work. Recording a complete history of digital forensics will benefit those who come after us. In the immortal words of George Santayana, "Those who cannot remember the past are condemned to repeat it" [14].

References

[1] American Society of Crime Laboratory Directors – Laboratory Accreditation Board, Garner, North Carolina (ascld-lab.org).

[2] ASTM International, ASTM E2678-09 Standard Guide for Education and Training in Computer Forensics, West Conshohocken, Pennsylvania (www.astm.org/Standards/E2678.htm), 2009.

[3] R. Downing, G-8 initiatives in high tech crime, presented at the *Asia-Pacific Conference on Cybercrime and Information Security*, 2002.

[4] Federal Bureau of Investigation, Innocent Images National Initiative, Washington, DC (www.fbi.gov/innocent.htm).

[5] Federal Judicial Center, Materials on Electronic Discovery: Civil Litigation, Federal Judicial Center Foundation, Washington, DC (www.fjc.gov/public/home.nsf/pages/196).

[6] K. Hafner and J. Markoff, *Cyberpunk: Outlaws and Hackers on the Computer Frontier*, Touchstone, New York, 1991.

[7] International Organization on Computer Evidence, G8 Proposed Principles for the Procedures Relating to Digital Evidence, Ottawa, Canada (ioce.org/core.php?ID=5), 2000.

[8] E. Iwata, Enron case could be the largest corporate investigation, *USA Today*, February 18, 2002.

[9] M. Mason, Congressional Testimony, Statement before the House Judiciary Committee, Federal Bureau of Investigation, Washington, DC (www.fbi.gov/congress/congress07/mason101707.htm), 2007.

[10] M. Noblett, Report of the Federal Bureau of Investigation on the development of forensic tools and examinations for data recovery from computer evidence, presented at the *Eleventh INTERPOL Forensic Science Symposium*, 1995.

[11] North Texas Regional Computer Forensics Laboratory, Dallas, Texas (www.ntrcfl.org/index.cfm).

[12] D. Parker, *Crime by Computer*, Scribner's, New York, 1976.

[13] RCFL National Program Office, Regional Computer Forensics Laboratory, Quantico, Virginia (rcfl.gov).

[14] G. Santayana, *Reason in Common Sense*, *Life of Reason, Volume 1*, Scribner's, New York, 1905.

[15] Scientific Working Group on Digital Evidence, Digital evidence: Standards and principles, *Forensic Science Communications*, vol. 2(2), 2000.

[16] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, Pocket Books, New York, 1990.

[17] The Charley Project, George Stanley Burdynski Jr., (www.charley project.org/cases/b/burdynski_george.html).

[18] The History of Computing Project, Timeline: Chronology of the history of computing, The History of Computing Foundation, Maurik, The Netherlands (www.thocp.net/timeline/timeline.htm), 2010.

[19] U.S. General Accounting Office, Crime Technology: Department of Defense Assistance to State and Local Law Enforcement Agencies, Letter Report GAO/GGD-00-14, Washington, DC (fas.org/irp/gao /ggd-00-014.htm), 1999.

[20] U.S. Secret Service, Electronic Crimes Task Forces and Working Groups, Washington, DC (www.secretservice.gov/ectf.shtml).

[21] C. Whitcomb, A historical perspective of digital evidence, *International Journal of Digital Evidence*, vol. 1(1), 2002.

Chapter 2

TOWARD A SCIENCE OF DIGITAL FORENSIC EVIDENCE EXAMINATION

Fred Cohen

Abstract Digital forensic evidence examination is not a normal science at this time. This paper discusses the important issue of moving toward a science of digital forensic evidence examination. It highlights key areas in which progress has to be made in order for digital forensic evidence examination to become a normal science.

Keywords: Digital forensic evidence examination, science

1. Introduction

Like almost every scientific endeavor, the examination of digital forensic evidence (DFE) started out somewhere between an art and a craft. People with special skills and knowledge leveraged their skill sets and knowledge to put forth notions about the meaning of DFE in the context of legal matters. While the court system greatly appreciates science and its role through expert testimony in providing probative information, the appreciation is substantially challenged by the lack of a scientific base. A scientific base includes a well-defined and well-understood body of knowledge, an underlying scientific methodology, an experimental basis, adequate peer-reviewed publications associated with professional societies, and all of the other things that go with normal science. As the volume and criticality of DFE has increased, there is an increasing recognition of the limitations of DFE, and more importantly, the limitations of the underlying science.

To clarify the notion of a science of DFE examination, it is instructive to examine the advancement of science in other disciplines. In most disciplines, a scientific methodology consists of four basic elements: (i) studying previous and current theories, methods and their experimental basis; (ii) identifying inconsistencies between current theories and repeatable experimental outcomes; (iii) hypothesizing new theories and performing experiments to test the new theories; and (iv) publishing the results. However, in an area where there is no pre-existing scientific infrastructure, a new theory, methodology, experimental basis, and perhaps even a new physics, have to be built from scratch. In the case of DFE examination, only one such attempt has been made so far [3], and this paper is substantially about that attempt.

2. The Call for Science in Forensics

The U.S. Supreme Court has spoken [8] and the National Research Council has concurred [6]. A rigorous scientific approach is needed for forensic evidence to warrant its use in courts in the United States. Much of the world is likely to follow this approach.

To a substantial extent, this call for science stems from some dramatic failures of forensics. For example, in the Madrid bombing case, the FBI declared that a fingerprint from the scene demonstrated the presence of an Oregon attorney. However, the attorney, after having been arrested, was clearly demonstrated to have been in another part of the world during the time in question. The side effect is that fingerprints are now being challenged as valid scientific evidence around the world [5].

A similar situation exists in cases where forensic examiners have done poor quality work and have testified in a number of cases, typically for the prosecution. The inability to effectively challenge evidence by these supposed experts using a scientific methodology and inquiry process is extremely disconcerting, all the more so because of the limits of human integrity. In case after case, when the details are examined, forensic evidence seems to come up short under competent challenges and close scrutiny. The solution is simple. Build and apply real science, and the truth will out.

3. Proposing a Science

This first attempt at proposing a science for DFE examination involves the creation and enumeration of elements of an epistemology and a physics of digital information, a model of the DFE examination process in the legal environment, and the interpretation of existing information, theory, and experimental results in the new model. This is the first attempt to create a scientific model for DFE examination, and a suitable name would be "the standard model." But of course, it will only become a standard to the extent that it is embraced by the community. Also, it would have to be adapted over with time as the digital forensic community comes to decide and adapt to the realities of the scientific method.

3.1 Epistemology for Digital Forensics

Epistemology studies the nature of knowledge, its presuppositions, foundations, extent, and validity. In the case of DFE examination, some basics may be reasonably assumed for the purposes of creating a science. The following are some of the epistemological issues that have already been identified.

Digital evidence consists entirely of sequences of binary values called bits. The physics of DFE is different from that of matter and energy, and thus the normal assumptions made with respect to how the world works do not apply – or do not apply in the same way – to DFE.

Two major differences are that DFE has observation without alteration and duplication without removal. Also, computational complexity limits what can be done with resources in a given time frame – one could say that the speed of light is "different" for DFE.

Physical evidence is very often transfer evidence and is sometimes trace evidence. In contrast, DFE is always trace evidence, but essentially never transfer evidence. Also, DFE is normally latent in nature in that it can only be observed through the use of tools. This implies a multitude of requirements surrounding DFE tools and their use.

In a "scientific" approach, the theories are not casual theories, but "scientific theories." That means that:

- Constructs are testable.
- Refutation can destroy a theory. Finite confirmations do not prove a theory; they can only confirm it.
- Scientific theories change slowly. Once theories are accepted, they only change – in very rare cases – because of dramatic changes in the understanding of the underlying physics.

The "theories" of DFE lead to a physics of digital information. Many of them are based on existing and widely-accepted mathematical knowledge. However, some are still conjectures from computer engineering, computer science, discrete mathematics, and related areas.

3.2 Information Physics

The physics of digital information is significantly different from that of the physical world. The differences are described in more detail elsewhere [3]. However, to get a sense of the differences, many of the underlying assumptions of the physical world, such as smoothness, continuous space, the notion of transfer, continuous time, and even the speed of light, are very different in the digital world. Indeed, the assumptions simply do not hold and the implications are, in some sense, profound.

Input sequences to digital systems produce outputs and state changes as a function of the previous state. To the extent that the state and outputs produce stored and/or captured bit sequences, these create traces of the event sequences that caused them. Thus, a trace may be defined as a set of bit sequences produced by the execution of a finite state machine.

We generally think of the physical world as a space that diverges with time, with any given initial conditions in history producing a wide variety of possible future outcomes. As a result, when looking at a physical trace, at least theoretically, it is possible to identify a unique historical event sequence that produced such a trace. But the digital space converges with time, so instead of the one-to-many relations seen in the physical world, there are many-to-one relations in the digital world. This means that a very large number of potentially different input sequences and initial states may produce identical traces (i.e., from subsequent states and sequences). Almost any digital trace could be the result of a large number of different historical event sequences, and the number of these sequences increases dramatically with the passage of time (i.e., execution of finite state machines). Thus, the traces from digital devices are not, in general, unique as to the input sequences that produced them.

Another less mathematical problem is the relationship between the unlimited granularity of the physical world in time and space and the finite granularity of the digital world in time and space. Because of this difference, a discontinuity exists at the interface between the physical and digital world. Minor differences are exaggerated near the discontinuity while major differences are ignored away from the discontinuity. The limited sensor and actuator capacity of devices that convert between the digital and physical world prevents the exchange of a variety of information that is potentially probative, and makes it much easier to create a variety of forgeries at the interface. This implies that input sequences may not directly demonstrate which non-digital events sequences may have produced them. As a result, additional effort is required to attribute traces to real-world causes and forgery is much easier in the digital space than in the physical space.

Viewing DFE as the result of processing by finite state machines inherently limits the potential utility of DFE examination for providing probative information about real-world events. DFE examiners must take the limitations into account when performing their examinations and when testifying about the results of the examinations. These limitations are directly due to the limits of DFE and the methodologies used to understand and work with it.

3.3 DFE Examination Model

The model of DFE examination is related to an overarching model of digital forensics [4]. It can be codified in mathematical terms as follows:

- **Laws:** $L : \{l_1, \ldots, l_n\},\ R : \{r_1, \ldots, r_m\},\ L \times R \rightarrow [F|T]$
- **Violations:** $L \times R \Rightarrow V$
- **Claims:** $E : [\{E_1, \ldots, E_o\}]$
- **Events:** $\forall e, e \in E^*$ that demonstrate claims $[\forall Ex \in E,\ Ex : (ex_1 \in E^*, \ldots, ex_p \in E^*)]$
- **Traces:** $T : (t_1, \ldots, t_q)$
- **Internal Consistency:** $C : T \times T \rightarrow [-1, 1]$
- **Demonstration Consistency:** $D : T \times E^* \rightarrow [-1, 1]$
- **Forensic Procedures:** $P : \{p_1, \ldots, p_n\}$, $\forall p \in P,\ p \rightarrow c \subset C,\ p \rightarrow d \subset D,\ p \rightarrow c \not\subset C,\ p \rightarrow d \not\subset D$
- **Resources:** $R : (T, \$, C, E)$
- **Schedule Sequence:** $S : (s1, s2, \ldots), \forall s \in S,\ s : (l \subset L, r \subset R, h \subset H, e \subset E, t \subset T, c \subset C, d \subset D, p \subset P, r \subset R, t, t')$

In essence, the legal claims constitute a set of runs through the elements of laws that produce violations. This can be conceptualized as a partially-ordered set (poset). The events and traces are entities that are evaluated to determine the outcome of the legal matter, and they form the basis for the claims that are demonstrated using the violation poset. If the events and traces are consistent with an unbroken path through the poset, a violation is indicated; if not, inadequate indications for a violation are present. If T and E are inconsistent with the poset, then they may act to sever all of the paths forming violations, in which case adequate basis may be present to definitively demonstrate that no such violation is justified. To the extent that T and E are internally or demonstrably inconsistent, C and D may be used to show that the evidence or the claims are less probative, or potentially even prevent the admission of elements of T and/or E into the matter.

The fundamental theorem of DFE examination in this model may be stated in relatively simple terms:

What is not consistent is not true.

DFE examination then consists largely of testing hypotheses related to the poset that form V as demonstrated by T and E in order to attempt to refute them by showing that they produce inconsistencies. This also implies some things about language and usage.

Consistency and inconsistency are demonstrated by logic and the theories associated with the physics of digital information. So, for example, given that a claim is based on an event e_1 causing a trace t_1, events and/or traces showing that t_1 happened before e_1 would be inconsistent with the claim of causality because information physics demands that cause precedes effect.

There are several consequences of this model related to: (i) the sizes of the model components; (ii) available computing power and its impact on thoroughness; (iii) limitations due to resources and schedules; (iv) limitations of available procedures; (v) legal limitations on what can be used and how; and (vi) probative versus prejudicial value and its relationship to consistency and related matters. In the example above, refutation is based on traces and events that may themselves be problematic. Thus, C and D are defined over the range [-1, 1].

In many cases, because of the limitations of DFE examinations as described here and elsewhere, more certainty is desired. Two general classes of methods exist to provide higher surety of DFE examination results: (i) identifying additional traces or procedures to gain additional demonstrations of consistency or inconsistency; and (ii) identifying redundant paths to prove hypotheses so that even if some paths are less certain or are eliminated, the overall hypotheses remain intact. These issues are covered by the model presented here.

3.4 Use of Defined Terms

No matter how many tests are performed, except for special cases, DFE examination results cannot prove a broad claim to be true [7]. The best that can be done is to show that the tests undertaken fail to refute the hypotheses and to show the extent to which the tests were thorough. This leads to the notion of what can reasonably be asserted as the most authoritative claim in [opposition] support of a hypothesis regarding DFE:

"The results of {the tests I did} were [in]consistent with {the hypotheses}."

To the extent that some of these statements are combined by logical reasoning, an overarching statement may be made with regard to the claims. This could be of the form:

> *"Based on {the basis}, the {traces and events} are [in]consistent with {identify the claim(s)}."*

Or in some cases, when this is true:

> *"In my examinations of {traces and events}, everything I found was consistent with {claims} and nothing I found was inconsistent with {claims}."*

On the other hand, a single refutation disproves a hypothesis. The least that can be reasonably said if such a refutation is identified is:

> *"The {procedures I performed} demonstrate that {traces and events} are [inconsistent with/refute] {the hypothesis}."*

Thus, the methodology underlying the science of DFE involves:

- Devising testable hypotheses (E)
- Testing the hypotheses against the evidence (T and E) using forensic procedures (P) and logic to determine Type C and D consistency by attempting to refute the hypotheses.
- Making careful and limited statements about the results of these tests, typically using wording such as that identified above.

The following are some wordings that may apply in other circumstances. Some of the more commonly misused ones are also identified, along with definitions appropriate for use by DFE examiners.

- **Suggest:** Imply as a possibility ("The evidence suggests ...") – calls to mind – propose a hypothesis or possible explanation.
- **Indicate:** A summary of a statement or statements or other content codified ("His statement indicates that ...") – a defined set of "indicators" are present and have, through some predefined methodology, and identified as such ("The presence of [...] smoke indicates ...").
- **Demonstrate:** Exemplify – show – establish the validity of – provide evidence for ("The reconstruction demonstrates that ...").
- **Correlate:** A statistical relation between two or more variables such that systematic changes in the value of one variable are accompanied by systematic changes in the other as shown by statistical studies ("Based on this statistical analysis, the use of the KKJ account is correlated (p=95%) with ...").

- **Match:** An exact duplicate ("These two documents have matching publication dates, page counts, ...").
- **Similar:** A correspondence or resemblance as defined by specified and measured quantities or qualities ("The 28 files were similar in that they all had syntax consistent with HTML, sizes under 1,000 bytes, ...").
- **Relate:** A defined and specified link ("The file system is related to FAT32 in that FAT32 was derived from ...").
- **Associate:** Make a logical or causal connection with basis provided ("I associate these bit sequences with program crashes because ...").

Through the careful use and consistent application of these terms, the field of DFE examination may move forward more quickly, and peer reviews could create a body of work that is meaningful across endeavors and time. However, if, the DFE community is inconsistent or if its peer review process fails to force compliance with the terminology, then DFE examination is unlikely to proceed as a normal science.

3.5 Tools of the Trade

As an area of science, DFE examination has a relatively small number of peer-reviewed and repeatable scientific experiments. Most experiments are of limited applicability and are not focused on building a fundamental understanding. They do not meet the standards of scientific rigor expected in other fields. They are oriented toward confirmation rather than refutation, which makes them dubious as science.

Furthermore, there is a methodological challenge associated with experiments for several reasons. DFE is latent and, therefore, experiments require tools. Of course, this means that the experiments are limited by the tools and, like any other area of science, the examiner must understand the limits of the tools in order to understand the limits of the experiments. This, in turn, leads to the need to have a methodology to evaluate tools. Without such a methodology, regardless of what the tools may indicate, the interpretation of the results is open to question.

A methodology for understanding tools might start with the development of an error model. Classical error models for digital systems [1, 2] may well be applicable, but their utility will not be known until they are applied in DFE examinations.

It is also important to understand how to calibrate and test tools, and to create systematic approaches for doing so. Calibration processes typically involve validation with known samples, which is readily done in most cases. The testing process typically involves verification of some sort, which, in the case of software, normally involves mathematical proofs or tests that verify the results against error models. Again, this is an area where DFE examination is lacking. Redundancy via the independent verification of results may provide an alternative in cases where no well-defined testing methodologies and practices are available.

Regardless of how "good" a tool is, it must be properly used, the results must be meaningfully interpreted, and the limits of the tool must be understood. This implies that the examiner must have knowledge, skills, education, training, and experience suited to the use of the tools they apply. DFE examination has few advanced students and teachers and, as a result, produces "niche experts," who are of limited utility. It has many niche experts who can potentially speak to very narrow domains and it has expert claimants who profess expertise beyond their actual knowledge, skills, education, training, and experience. Indeed, DFE examination as a field has too few "real experts" at this time.

3.6 Presentation

Another major issue with tools today is how they present results, both in support of the examination process and when the results of examinations are presented in reports or before judges and juries.

Presentation is intimately tied to, but not directly part of, examination. Because DFE is latent, presentation will always be an issue. For the examiner, the results of the experiments must be presented using tools. For the jury, presentation is again fundamental to understanding the evidence and the examination results. For the judge, the same is true to evaluate admissibility. For the opposition, presentation is just as critical to evaluating expert reports,. Today, however, there is no standard for even presenting the most common representations of DFE. Even something as simple as presenting a text file is fraught with potential errors.

Different ways of presenting the same information can lead to different interpretations and outcomes. Consider this simple example:

- Plaintiff's sworn statements are inconsistent with the evidence.

- If Plaintiff's sworn statements are to be believed, then the evidence refutes Plaintiff's claims.

- If the evidence is to be taken at face value, then it refutes Plaintiff's sworn statements.

The first of these statements encompasses the other two. The second statement appears to say that one can assume that the plaintiff is telling the truth, but the evidence does not support the claim. The third statement appears to say that the plaintiff is lying.

Technical presentation errors are also problematic. For example, the digit "O" and the letter "0" are almost indistinguishable, as are the digit "l" and the letter "1." Spaces at the ends of lines, and the differences between a leading tab, a leading space followed by a tab, and leading spaces cannot be discerned in normal printouts. As an aside, the nature of the problem becomes very clear if the reader failed to notice that the numbers "0" and "1" and the letters "O" and "l" are interchanged above.

When examining the output from widely-used and trusted tools, the presentation produced by the tools often fails to aid the examiner in seeing these sorts of differences. In case after case and in tool after tool, differences that might allow the examiner to detect inconsistencies go unseen and, thus, the inconsistencies are commonly missed. Even something as simple as a forensic font would largely alleviate these problems, but this notion was only first introduced in late 2009.

Clearly, presentation is fundamental to the advancement of a science of DFE examination. Also, presentation is critical to the effective use of tools upon which essentially all of digital forensics depends.

4. State of the Science

The principal elements of DFE examination are analysis, interpretation, attribution and reconstruction. While this categorization of the discipline may be somewhat arbitrary, it is sensible in that each of these elements, while not entirely distinct from one another, encompasses different techniques that may require different expertise and tools.

4.1 Analysis

Analysis engages techniques that provide definitive answers to specific questions. Existing analytical techniques are limited in number and in the nature of the definitive results they produce. The techniques are computational in nature with defined complexity and defined input and output limitations, and their accuracy can be verified (at least theoretically). Methods that do not meet these criteria should not be called analysis.

Analysis starts with a bag of bits. Redundancy in the bag of bits confirms or refutes hypotheses about what the bag of bits is. Through analysis, features and characteristics are detected, symbol sets are identified, trace typing is undertaken, content is parsed, normalized and elucidated, and indicators are analyzed. Any of these may return the examiner to a bag of bits, especially if it produces inconsistencies that destroy the chain of analytical results supporting the current hypotheses about the bag of bits.

During analysis, characteristics and features are analyzed for consistency, traces are ordered and out-of-order entries are detected, sourcing and travel patterns are identified, consistency is checked across related records, anchor events are used for external validation, time differentials and jitter are considered, and all of these are compared with hypotheses in order to identify consistency or inconsistency. Also, sieves are constructed and items are counted, derived traces are formed for analytical convenience, counts are made of various features and characteristics of interest to the examiner, mechanisms are combined and the resulting errors identified and mitigated, and results are verified by independent means where feasible. Additionally, intentionally-hidden items of interest are sought, content placed in hard-to-find locations is found, steganographic and other transformed content are identified and inverted, recursive embedded languages are parsed, and indicators are identified and sought relative to the issues at hand.

In these analytical processes, automated and human cognitive methods are combined to identify potential Type C and D consistencies and inconsistencies either automatically or through DFE examiner interaction. All of these processes are limited by the available forensic procedures (P) and their computational complexity, which impacts and is impacted by resources.

4.2 Interpretation

Traces, events, claims, and analytical results must be interpreted in the context of the legal matter in order to be meaningfully examined. Interpretation involves selecting among alternative explanations (hypotheses). Different interpretation methods are used for different circumstances, with one major difference being the treatment of structured and unstructured traces.

Over-interpretation by examiners is common. For example, terminology is misused, conclusions are drawn in excess of what is supported by the data, the terms "match" and "correlate" are overused, and the details of the basis are often left unspecified.

Special care must be taken in making statistical claims. In addition to simple interpretation errors (e.g., percentages that add up to more than 100%), claims often ignore data that is not present, conceal assumptions related to random stochastic processes and other similar things, and assert precision exceeding accuracy. This is particularly problematic in digital systems, where 33.333% results from an experiment with only 9 samples. At best, there is a little less than one digit of accuracy, so this precision is misleading. The examiner must interpret the output of analyses and present information in clear terms in order to properly interpret results for presentation – 3 of 9, 1/3 or 33% are all valid, but 33.333% is not.

Tools commonly interpret traces as well, and to the extent that they do so, these interpretations make assumptions, often without basis. The interpretations often present false results based on the assumptions and result in presentations and other depictions that may mislead the examiner as well as the judge and jury. No methodology currently exists for evaluating interpretation by tools, although a wide range of cognitive errors are known to cause incorrect or imprecise interpretation.

Interpretation includes the identification and explanation of missing traces and their implications to the matter at hand. While some experts may know the traces that are commonly present in some situations, there is neither a widely available library of what constitutes "normal" behavior in an operating environment nor a basis for comparison in the existing literature.

While redundancy may be used to mitigate interpretation errors, interpretation is highly subject to individual variations. Proper interpretation is limited and couched in terms of its accuracy and applicability, including the things that the examiner does not know. It is important to enforce careful word usage for technical terms in professional publications and in legal venues. How can one claim to be an expert when one does not even use the published and accepted terms of the scientific and technical community?

There is also a tendency to create casual theories and embrace them as if they were something more. To be clear, the theories of DFE examination are the results from information physics, the assumptions of the model, and the foundational logic and results of computer science and engineering. Any other theory, unless and until it passes peer review and is reconciled with the existing theory, is only a hypothesis at best.

Examiners, like all people, make cognitive errors during the interpretation process. The examiner would be well served to examine the literature in this area and apply it to maintain clarity regarding his/her own as well as others' potential for making mistakes. Indeed, it is unwise to believe what one thinks is right unless one really knows. What some reviewers call "put a stake in the ground" should be shunned in the DFE community in favor of properly identifying the limitations and not overstating a case. Peer review of examination results, particularly interpretations, should be sought in every case and for every report.

As a working assumption, the DFE examiner should assume that each system and situation is in fact different from others. Experience is a great teacher, but it can be misleading. Care must be exercised in not over-interpreting the data. And when the interpretation is unclear, reconstruction is a viable and worthwhile approach for gaining additional data. All interpretation should also be made within the context of information physics, and with the working assumption that every interpretation will be tested against information physics and refuted if it is inconsistent.

Events are also subject to interpretation because words are interpreted by examiners. Cognitive errors in interpretation are common and, since examiners assume the context of their knowledge and words mean different things to different people, the careful interpretation and close scrutiny undertaken by an examiner may be viewed as being "picky" by others; but this is the nature of interpretation in the legal environment. Furthermore, events are often difficult to interpret and leave many possible interpretations, particularly in light of the lack of specific technical knowledge by the individuals who offer the information related to the events.

Events should presumably be viewed through the lens of information physics. Claims are often inconsistent with and are, thus, refuted by the careful use of physics. As an example, causality is tricky because of the requirement for time ordering and the fact that complexity limits actions in the digital space; this results in a poset of causal relationships. Reviewing information physics for events may be helpful, but being thorough in this regard is problematic because of resource limitations and the lack of apparent progress when the examiner is thinking about the issues in light of other considerations. At this time, it is not possible to automate much of this examination and most DFE examiners lack the time or knowledge to do a thorough (or even thoughtful review) against the full set of theories.

Resources limit interpretation and tight schedules may prohibit contemplation of the issues in many cases. Computational resources and costs may be prohibitive for certain procedures, especially when a large volume of traces are involved. Available traces may change with time and legal actions as well.

Statements made and documents created by examiners are often unnecessarily interpretive. Careful wording is highly desirable in written and verbal communications. There are few "standard wordings" for DFE examination, and examiners often make statements that end up being wrong, but that could have been accurate had they been stated differently. For example:

"I found X B's in File Q"

is true even if there are more than X B's present, because, presumably, the examiner did not find the other ones. On the other hand, the interpretation:

"There were X B's in File Q"

is potentially problematic given the same circumstance. It is a good idea to minimize interpretation and favor statements of fact wherever possible.

Similarity is almost always interpretive, in large part because DFE examination lacks adequate similarity metrics or criteria, an experimental basis for many similarity claims, and techniques for detecting similarity in many cases. Still, examiners have the tendency to use the term "similar" when it cannot be backed up with facts. Similar problems are often indicated and commonly associated with the suggestion of matches relating associated correlations. Clearly these uses are unclear and imprecise, and the lack of clarity and precision during use, while perhaps suggestive to the untrained reader, should be readily identified as indicative of a lack of knowledge, education, and training in DFE examination.

Assumptions underlying interpretations are critical, but they are often not detailed in statements and reports. Thus, the statements and reports lack the basis for the interpretations. It may be hard to identify all the assumptions, but confirming/refuting assumptions and hypotheses is the process that should be relied upon in interpretation; therefore, identifying and presenting them is important to obtaining the right answer.

Presentation is also an interpretive function that drives every aspect of DFE examination. It acts within the process of examination to skew the approaches and procedures used by the examiner. The presentation provided by a tool is the only thing the examiner has to rely on to understand the latent traces. Examiners must understand that they are interpreting the output of tools and that the outputs and interpretations are fraught with the potential (often realized) for producing cognitive faults, leading to failures that may significantly impact examination results and process.

4.3 Attribution

Correlation is not causality; before is not because. On the other hand, a lack of correlation certainly throws considerable doubt on the notion of causality, and the cause had better be before the effect.

Attribution of actions to actors is centered on the notion of causality, to the point where they are, in essence, inseparable. The fundamental assumption of causation in the digital world is:

> *Traces come about by the execution of finite state automata that follow the physics of the digital world.*

The physics of the digital world is useful in assessing attribution claims. For example, A caused B implies that A came before B. How long before is determined by the digital version of the speed of light. Computational complexity, the performance levels of devices, the physical speeds associated with the devices and their components, and the mechanisms available for storage, transport and processing, all place limits on the interval of time between the cause and the effect.

Statistics do not apply in most cases relating to attribution because the past is what it is; there is no "might have been" or likelihood that some particular thing happened. Either it happened or it did not happen.

Establishing a causal chain is non-trivial as well. It often involves redundant records and almost never eliminates all other possibilities. But it may provide a seemingly overwhelming body of information that is consistent with the hypothesis of attribution, and no information available may lead to inconsistencies with a hypothesis. This becomes a compelling argument for a judge or jury, even if the examiner never claims it to be definitive.

Finite state machines are highly predictable because driving state and input to output leads to the same answer every time in almost all cases. However, finite state machines converge with time (while the real world diverges), so where simulation may produce identical outputs, reversing time does not give a unique answer. Therefore, in the digital world, convergence implies that many paths lead to the same traces. In addition, because sensors that change physical inputs to digital outputs are highly nonlinear, small differences are expanded near a nonlinearity while large differences are reduced far from a nonlinearity. Thus, the interface tends to break the digital perfection of even forward-driven causal chains.

Attributing actions to human actors is even more problematic. Authentication methods are of limited value. Most biometrics are not good for identification, but rather only for selecting a known individual out of a group of a few thousand known individuals when deception is not in use. Something the user has can be taken or exploited; something the user knows can be known by others; something the user can do can be done by others.

DFE can almost never put a person at a keyboard at a time, although other events may be able to help. And even if the user is at the keyboard, it does not necessarily mean that he/she was in control of the computer. Behavioral attribution may use words and word sequences, commands that are executed and usage patterns, keyboard use and timing patterns, etc., but all of these fail under deception. Also, they are limited to picking known individuals out of small known groups and they have significant error rates even under non-stringent test conditions.

Device authentication and attribution may be accomplished using various indicators (e.g., operating environment identification data, device identifiers, and known behaviors of programs and mechanisms). While most of these are readily subverted by deception, some have properties that make them difficult to forge. Redundant traces may be applied to reduce the impact of deception.

Attribution of damage to parties is also important, e.g., to establish threshold requirements for criminal charges. Actual damages are typically divided into: (i) physical damage; (ii) conversion to use by another party; (iii) deprivation of utility to the owner, which is often the key DFE issue; and (iv) lost value or lost rights (e.g., disclosure of trade secrets or release of pre-patent data). If attribution can be done of the cause to effect, computation methods are available to identify the extent of the deprivation. This may include things such as the cost per usage of electricity, cost in reduced lifetime of equipment, cost in demonstrable lost business, and cost in reduced life of equipment.

4.4 Reconstruction

Driving time backwards is one approach to reconstruction, but information physics shows that this is problematic. The lack of adequate traces over time leads to very large envelopes of possible histories, which make it almost impossible to tell what was "original." In addition, theft may not be identifiable, travel time and jitter produce ordering uncertainties, and reversing time in a unique manner through homing sequences is impossible. Error accumulation also leads to large expansions when reversing time. The list goes on and on, which leads to the alternative of experimental demonstration of operation in the forward direction.

Experimental demonstration of operation in the forward direction is a form of reconstruction that can be used to test hypotheses. As such, it can confirm, refute, or be unrevealing. The basic methodology deals with constructed traces (C-traces) and original traces (O-traces). Similarity measures are used to define, in advance, the criteria for identifying sets of

C-traces mapped to O-traces, and the implications of outcome class sets. The examiner creates reconstruction(s) based on hypotheses, generates C-traces, and compares the C-traces to the O-traces to confirm or refute hypotheses.

5. Toward a Normal Science

Based on the discussion above, the following statements can be made today based on the science of DFE examination:

"I did X and observed Y."

"I [did not find/found] X in Y."

"X is [in]consistent with the claim Y because..."

"X [suggests/indicates/demonstrates/correlates with/matches/is similar to/relates to/associates with] Y because..."

If examinations are properly undertaken, each of these can have a sound basis with the proper scientific underpinnings. Unfortunately, the current set of methodologies, processes and procedures are limited in terms of their validity, testability, reliability, calibration, and basis. Additionally, there is a lack of strong agreement within the DFE community about many aspects of the science as a whole. While most of the results are peer reviewed and accepted within individual communities, several problems exist:

- The overall collection of results (as a body of science) is not recognized as such.
- The unifying methodology expressed with regard to the application of information physics to determine consistency is gaining acceptance very slowly.
- Models that are currently in use are used for various limited purposes, and are not widely adopted.
- Procedures and their results are limited and are not formalized or standardized.
- Tools and processes are only explored to a limited extent, with notions of completeness and thoroughness just beginning to be defined.
- Error models have not been adequately applied from other mature fields. The sources and magnitudes of uncertainty are poorly defined, and confidence intervals simply do not exist.

In most cases, the honest and knowledgeable examiner is largely limited to the most basic:

"I did X and observed Y,"

with the observation being typically limited to:

"I [did not find/found] X in Y."

While these are powerful statements that are appropriately used in place of other less sound statements, they are a long way from the level of science that DFE examination has the potential to achieve.

6. Conclusions

DFE examination is not operating as "normal science" today. While there is a scientific basis for many activities involved in DFE examinations, a widespread consensus and common methodologies are still lacking. The foundation for scientific theories exists, but little attention is paid to testing the theories and developing the science. To successfully make the transition to a normal science, the DFE community will have to ask and answer several questions:

- What well-defined and consistent terms should be used?
- What well-understood epistemology should be used?
- What theories and methodologies should be chosen?
- What strong experimental foundations should be built?
- What agreed-upon physics should be used and how should it be formulated?
- How could community consensus be built?
- Should the path outlined here be embraced?
- If not, what is the best path?

The view of this paper is that there exists at least one description of a reasonably comprehensive scientific foundation underlying DFE examination. Regardless of the problems and limits of the foundation, it is a place to start building a normal science and advancing the field. As the field matures, normal science is almost inevitable, but the normalization process is only just beginning. A community consensus is highly desired, and this paper supports and anticipates such consensus in the near future.

References

[1] A. Avizienis, J. Laprie, B. Randell and C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, *IEEE Transactions on Dependable and Secure Computing*, vol. 1(1), pp. 11–33, 2004.

[2] M. Breuer and A. Friedman, *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, Rockville, Maryland, 1981.

[3] F. Cohen, *Digital Forensic Evidence Examination*, ASP Press, Livermore, California, 2009.

[4] F. Cohen, Two models of digital forensics examination, *Proceedings of the Fourth International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 42–53, 2009.

[5] G. Fine, Statement of Glenn A. Fine, Inspector General, U.S. Department of Justice before the House Committee on the Judiciary, Subcommittee on Crime, Terrorism and Homeland Security concerning Section 1001 of the USA Patriot Act, U.S. Department of Justice, Washington, DC (www.justice.gov/oig/testimony/0505b.htm), May 10, 2005.

[6] National Research Council of the National Academies, *Strengthening Forensic Science in the United States: A Path Forward*, National Academies Press, Washington, DC, 2009.

[7] K. Popper, *The Logic of Scientific Discovery*, Hutchins, London, United Kingdom, 1959.

[8] U.S. Supreme Court, Daubert v. Merrell Dow Pharmaceuticals, Inc., *United States Reports*, vol. 509, pp. 579–601, 1983.

Chapter 3

USING A LOCAL SEARCH WARRANT TO ACQUIRE EVIDENCE STORED OVERSEAS VIA THE INTERNET

Kenny Wang

Abstract This paper argues that a search warrant issued by a local court does not have the power to search and seize digital evidence stored overseas but accessible via the Internet. Based on the fact that digital evidence can be altered or erased in a very short time, two scenarios are presented to illustrate the lack of power of a local search warrant to acquire digital evidence overseas. Two solutions are presented to overcome the shortcomings of a local search warrant. These solutions can assist law enforcement agencies around the world in searching and seizing digital evidence stored overseas with speed and accuracy, and in addressing court challenges regarding the admissibility and potential illegality of this evidence.

Keywords: Search warrant, remote cross-border search, webmail

1. Introduction

Computer-related crimes are not a new type of crime in Hong Kong. Hong Kong has had computer crime laws since 1993 [10], but the popularity of information and communications technology in recent years has contributed to an increase in the number of computer-related crimes.

For most individuals, a very important function of computers is to store data. Data used by a computer can be stored in various media such as flash drives, hard disks, compact disks and magnetic tapes. Data can also be stored remotely on a server, which may be situated on the same floor as one's office, or on another floor, another building or even another country. Dealing with data that is stored in another country and accessible only via the Internet (web storage) is more complicated in terms of technology and legal jurisdiction, but it is a growing trend. According to

Hirst [13], "[t]he Internet has an international geographically-borderless nature ... you can find things in it without knowing where they are. The Internet is ambient – nowhere in particular and everywhere at once."

Many law enforcement agencies around the world, including in Hong Kong, have stood up special units to deal with computer-related crimes. Although law enforcement agencies aggressively investigate computer crimes, criminals often avoid detection by utilizing computer technologies. To win these battles, law enforcement needs traditional as well as advanced procedures and tools.

The search warrant is a powerful traditional tool. It empowers law enforcement agents to gain entry into a suspect's premises and to search and seize any items in the premises that may constitute evidence of a crime, including computers. It is increasingly common for criminals to store their information overseas and to access it over the Internet. Is it possible for a law enforcement agent to use a local search warrant to search and seize digital information stored overseas but accessible via the Internet?

This paper analyzes the possibility of using a local search warrant, i.e., a search warrant issued in Hong Kong, to search and seize evidence stored on an overseas computer but accessible via the Internet. More specifically, it focuses on the notion of a "remote cross-border search," which is defined as using a computer within the territory of a country to access and examine data physically stored outside of its boundary [1]. The paper also examines the connection between remote cross-border search and extraterritorial jurisdiction. The goal is to help prevent digital evidence from being excluded by a judge because it was obtained illegally. This not only wastes resources but may also expose law enforcement agents to civil or criminal proceedings.

The paper presents two scenarios and analyzes them in the context of various statutes, case law and legal research to determine whether or not a local search warrant can successfully acquire digital evidence that is stored overseas without raising challenges from defense counsel.

2. Crime Scene

Suppose that a law enforcement agent in Hong Kong is in possession of a search warrant issued by a magistrate. The search warrant empowers him and his colleagues to gain entry into a premises and to search and seize items relevant to the investigation. The target is suspected to have committed fraud. The suspect is a computer expert so it is very likely that his computer contains considerable information related to the case.

The D-Day arrives. The law enforcement agent knocks on the door of the suspect's home. The suspect answers the door. The agent arrests the suspect, cautions the suspect, presents the search warrant, explains it to him and starts the search. The agent sees a notebook computer in the living room. Upon examining the computer, he discovers records of bank accounts and transactions that may be related to the investigation. The agent has found some valuable evidence, but there is more. What is this other evidence?

We consider two scenarios. In the first scenario, the law enforcement agent sees a webmail login page on the computer screen. The suspect has already entered his username and password. All that remains to be done is to hit "Enter" on the keyboard.

In the second scenario, the agent finds a piece of paper on the suspect's desk. The paper contains the name of a webmail service provider (WSP), a username and a password. The agent believes that the username and password belong to the suspect's webmail account, but the suspect remains silent when questioned.

We assume that the WSP is in the United States and some e-mail messages in the webmail account may be vital to the successful prosecution of the suspect. How does the law enforcement agent legally obtain the e-mail evidence so that it is admissible in a Hong Kong court?

3. Scenario 1

This scenario involves jurisdictional issues between Hong Kong and the United States. The suspect has a webmail account with a U.S.-based WSP. The WSP maintains numerous mail servers that store millions, possibly billions, of e-mail messages belonging to users from around the world. All the mail servers are physically located in the United States; thus, the servers and their contents are under U.S. legal jurisdiction. The Hong Kong law enforcement agent wants all relevant information pertaining to the webmail account such as registration details and e-mail contents, but the information is in the United States. How can the law enforcement agent legally acquire the information from overseas?

3.1 Traditional Method

The traditional method for handling the case is to rely on mutual legal assistance (MLA). Two governments sign an MLA whereby each guarantees to provide support to the other on criminal matters. Hong Kong and the United States have an MLA agreement [11]. The agreement includes "executing requests for search and seizure." This makes it possible for the U.S. authority that handles the MLA (U.S. Department of Justice (USDoJ)) to search the WSP in the United States and seize the relevant information (including the e-mails) on behalf of the Hong Kong Department of Justice (HKDoJ). Under MLA, the USDoJ obtains the registration details and e-mail messages belonging to the suspect and sends the information to the HKDoJ, who then passes it to the requesting law enforcement agency.

The MLA arrangement is feasible but slow – standard MLA procedures may take weeks, if not months, to complete. During this time, it is very likely that the digital evidence and possibly the suspect himself have vanished. The suspect, who normally can only be detained up to 48 hours after his arrest, would have been released long ago and could use any Internet-ready computer to delete all the evidence in his webmail account. Even if the suspect was charged and remanded in prison, he could arrange for someone else to access his account and delete his e-mails. The suspect would be confident that, by the time the MLA procedures are completed, his deleted e-mails would have been gone for weeks or even months and would not be recoverable. The point is that no matter how efficient MLA procedures are, law enforcement agents can never be sure that they will be able to seize digital evidence stored outside their jurisdiction [1].

Is there any way that the e-mail contents stored overseas can be acquired accurately and with speed so that the evidence is not lost?

3.2 Direct Method

The direct method is to access the suspect's webmail account in the United States simply by clicking the "Enter" key on the keyboard. Having accessed the webmail account, all of the suspect's e-mail messages can be downloaded to his computer in Hong Kong so that they come under Hong Kong jurisdiction. Does the local search warrant empower the law enforcement agent in Hong Kong to access digital evidence stored overseas via the Internet?

The answer is in the negative. Before the "Enter" key is depressed and the e-mail messages are downloaded, the messages are still in the United States. The messages would not have been delivered to the suspect's computer in Hong Kong had the download request not been made. As a result, the digital evidence obtained by the law enforcement agent can be regarded as illegally obtained and subject to challenge in court. Could hitting "Enter" on the keyboard and clicking the mouse a few times have such an impact?

The answer is in the affirmative. It is similar to a situation where law enforcement agents execute a search warrant on a subsidiary office in a different location from the main office. During their search, the agents discover that the subsidiary has a computer network that links to the main office. If a piece of digital evidence stored on the server of the main office is accessible via the computer network at the subsidiary, what should the agents do? Of course, they should apply for another search warrant to search the main office. The first search warrant does not empower them to obtain digital evidence stored in the main office. As a result, it is unlikely that a local search warrant in Hong Kong would permit an agent to search and seize e-mail messages stored overseas.

Johnson and Post [14] argue that the emergence of the Internet has destroyed not only the power of a government to regulate online behavior but also the legitimacy of a sovereign to "regulate global phenomena." The USDoJ [18] admits that although digital evidence seized remotely from one district to another district may be admitted by U.S. courts, it is a different matter altogether when the evidence is located outside the United States; moreover, remotely searching data stored outside the country is a complicated matter. Clearly, the USDoJ has some reservations regarding the validity of remote cross-border searches.

Graham [12] comments that for certain cyber crimes, such as child pornography and hate speech, an international enforcement jurisdiction is justified. Dauterman [7] argues that a state can assert "reasonable jurisdiction" over a person who commits an offense outside the state as long as it causes harmful effects in the state and if there is a substantial connection between the person and the state, the latter should be reasonable to claim extraterritorial jurisdiction. The Arkansas v. Kirwan case [16] demonstrates that U.S. courts could claim jurisdiction as long as a suspect's conduct or the result of his conduct occurred within the state. Nevertheless, it is doubtful that the e-mail content in our scenario fulfills any of the requirements for extraterritorial jurisdiction. This is because the offense involved is not a universal crime. Also, even if the suspect has a substantial connection with Hong Kong, it would be hard to prove that storing e-mail overseas causes harmful effects in Hong Kong. Hiding criminal evidence may have a harmful effect on the case, but not to the Hong Kong public.

Some U.S. legal scholars favor remote cross-border searches. There is a view that countries like the U.S. long for the unilateral power of remote cross-border searches without assistance from or even the acknowledgement of the country where the data is stored [1]. The United States v. Gorshkov [3] and United States v. Ivanov [19] cases show that, when the need arises, U.S. courts are willing to try overseas cyber criminals even though they were not in a U.S. jurisdiction when they committed their crimes, and the courts may admit the digital evidence even if it is obtained by a unilateral remote cross-border search [4]. Although these two cases were successfully prosecuted, there is a comment that the Gorshkov case cannot give the "conceptual basis" for the legal issue of cross-border search and seizure because it implies that any country that suffers a crime originating from another country can invade the country's sovereignty by remotely searching and seizing property physically located in that country [4]. Also, there is the possibility that the country that suffered the remote cross-border search could retaliate. In fact, Russia's Federal Security Service brought criminal proceedings against the FBI agent in the Gorshkov case who was responsible for accessing the computers in Russia [5].

Based on the cases cited, it appears that the U.S. judiciary has never denied the possibility of extraterritorial jurisdiction; thus, the validity of remote cross-border search may not face many legal challenges in the United States. Goldsmith [9] even argues that cross-border search is a necessary tool against cyber crime, that there are precedents, and that it is inevitable in order to accommodate new technology. On the other hand, Brenner and Schwerha [4] maintain that it is not clear if U.S. law enforcement agents can "lawfully" use computers in the United States to seize digital evidence stored overseas. They also state that the only certainty is that an agent who has conducted such a seizure can claim that, because the digital evidence could be destroyed or moved, he had to obtain the evidence quickly, possibly even without a search warrant [4]. Of course, it is up to the court to accept this claim.

The 2001 Cybercrime Act of Australia grants power to law enforcement agents to "operate electronic equipment at the warrant premises to access data (including data not held at the premises)" if the data might be relevant to the case and the equipment can be operated without damaging it. This act appears to "legalize" remote cross-border searches in other countries, but it does not say whether the Australian authority allows other countries to do the same to computers on Australian soil.

It is important to note that legislation and arguments that favor remote cross-border searches can lead to a situation where a law enforcement agency in any country that has Internet connectivity would have jurisdiction to lawfully access any data stored on the Internet. Such a situation would likely result in chaos because every country could claim legal jurisdiction to Internet servers around the world. This would severely restrict the freedom of speech because no matter where a person voices his views on the Internet he may have committed an offense (usually political) in some part of the world. The irony is that many in the United States claim that the Internet is an ideal place for freedom of speech, but now it appears that any government in the world can control *any* content on the Internet. In fact, if this were to be the case, the U.S. would suffer because any law enforcement agency in the world – from Canada, France, China, Peru, even Iran – could claim jurisdiction over all U.S.-based servers with Internet connections.

In the international arena, law enforcement agents from one country cannot exercise their power in the territory of a second country except with the consent of the government of the second country [1]. Berman [2] comments that a target state of a remote cross-border search may feel that the extraterritorial investigations by other states threaten its citizens and may, therefore, impose measures such as privacy protection to limit the scope of investigations or even bar the investigations. In general, a unilateral remote cross-border search violates customary international law [1].

Recognizing the problems associated with MLA, the Council of Europe has suggested various measures for the smooth transfer of digital evidence between states. In 2001, the Council of Europe's Convention on Cybercrime [6] recommended that a speedy MLA such as fax or e-mail should be used to facilitate the rapid preservation of digital evidence. Article 32 of the Convention on Cybercrime legalizes remote cross-border searches of publicly-available data and other data with the consent of the local authority. Nevertheless, the data in our scenario is not publicly available and seeking WSP content may be fruitless. The WSP does not have any legal obligation to comply with a request from overseas and the WSP is not the owner of the data, so it may not even have the authority to give consent to an overseas authority. Thus, it appears that Article 32 does not provide much assistance in our scenario.

Based on our analysis, the problems related to traditional MLA remain. The Hong Kong law enforcement agent may not unilaterally conduct a remote cross-border search on the suspect's webmail account to avoid the embarrassment that the evidence obtained is inadmissible because an offense was committed by the law enforcement agent during its extraction [21].

Is there a better approach? We analyze the second scenario before presenting two solutions.

4. Scenario 2

This scenario differs from the first scenario in that the law enforcement agent has the username and password of the suspect's webmail account. But the problem is that the search warrant is not valid unless it has an extraterritorial effect. Moreover, it is not feasible for the agent to apply for another search warrant because there is nothing local to search.

Conducting a remote cross-border search and seizure using the suspect's username and password is similar to using a key found in the suspect's premises to open the suspect's safe deposit box in a bank. The matter is more complicated if the safe deposit box is located overseas. It is inconceivable that the law enforcement agent would travel to the country where the deposit box is located, open the deposit box and bring its contents back to Hong Kong. The agent does not have the power to search and seize items in another country. Although it can be argued in this scenario that the law enforcement agent has not left his jurisdiction and is, therefore, not acting without authority, it is doubtful that the evidence acquired via the Internet would be admissible in court.

5. Two Solutions

Based on the preceding analysis, we should recognize that, although it is technically feasible to conduct a remote cross-border search and seizure, it is an exception rather than a rule for the court to accept the digital evidence. We present two solutions. Note that these solutions are not related to the illegality of the contents (e.g., pornography or hate speech) as in our hypothetical scenarios. Indeed, there is nothing wrong with the e-mail content *per se*. It is just that the suspect has stored his criminal evidence abroad hoping to take advantage of the jurisdictional restrictions imposed on Hong Kong law enforcement.

5.1 Solution 1

The first solution involves duress, but it has been implemented. The Bank of Nova Scotia case [17] demonstrates that, if necessary, U.S. courts are willing to compel foreign banks with U.S. branches to produce bank records held outside the United States that are subject to foreign jurisdiction. Similarly, a magistrate or judge in Hong Kong could issue a local search warrant to search the Hong Kong subsidiary of the WSP (if one exists), and compel it to produce information about the suspect's webmail account and the contents of all e-mail messages. The rationale is that the webmail account, no matter where it is physically located, is treated as a local webmail account. As long as the owner of the webmail account can access it in Hong Kong, courts would ignore where the webmail account and its contents reside and compel the local subsidiary of the WSP to disclose everything about the account.

Because the evidence is given by the Hong Kong subsidiary, it is treated as evidence obtained locally, so the jurisdictional issue does not arise. While this solution appears to eliminate the jurisdictional problem, the irony is that it turns a blind eye to the issue of extraterritorial jurisdiction. As a result, multinational companies such as Yahoo or Microsoft would suffer.

Another important point regarding this solution is that the country that uses it must have strong economic bargaining power that would force multinationals to comply; otherwise the companies could pull out and move to a "friendlier" country. Nevertheless, this solution sets aside the problem of extraterritorial jurisdiction and enables a local search warrant to acquire digital evidence stored outside the jurisdiction.

5.2 Solution 2

The second solution attempts to strike a balance between speed and jurisdictional issues; also, it emphasizes the integrity of evidence. In this solution, the Hong Kong law enforcement agent may search and seize the digital evidence immediately using the suspect's computer or using the information on the piece of paper that contains the suspect's webmail username and password. Having seized the evidence, the law enforcement agent should contact the WSP in the United States by fax or e-mail to request the immediate preservation of evidence in the suspect's webmail account pending MLA approval. Meanwhile, the agent should issue a formal MLA request [20]. This solution is different from the one described in Article 32 of the Convention on Cybercrime because it does not require "lawful and voluntary consent" by the WSP. Note that the WSP will only preserve the digital evidence; it will not surrender the evidence to the Hong Kong agent until and unless ordered to do so by a U.S. court under the MLA request.

This solution preserves digital evidence before it can be deleted. Of course, if the evidence from the WSP is the same as that seized in Hong Kong, the evidence from the WSP can be presented to the court directly. However, in the unlikely event that the evidence from the WSP is different from the evidence seized in Hong Kong, more weight should be given to the evidence from the WSP.

This solution is attractive because it conforms with MLA practices. Also, the evidence is preserved speedily and its integrity cannot be questioned because the preservation is done by the WSP, not by the Hong Kong law enforcement agent.

6. Conclusions

The two solutions presented to overcome the shortcomings of a local search warrant are far from perfect. As long as there are cross-border searches on the Internet, there will be debates on how to maintain law and order in cyberspace without infringing the jurisdiction of another country and the privacy of its citizens. While it is useful to consider streamlining current legal procedures for acquiring digital evidence overseas, in the long term it is important to deal with this issue in the context of MLA agreements. This means that countries may have to negotiate and agree on joint cross-border searches over the Internet. Joint operations require good intelligence and coordination or they could lead to disastrous consequence as in Operation Ore, where thousands of British men were suspected of accessing child porn websites based on credit card payments, when, in fact, some of them were victims of credit card fraud [15]. The European Union provides a good example of transnational cooperation. The European Union has established a border-free travel zone based on its confidence in the border control procedures of its member states [8]; similar confidence could be placed on its member law enforcement agencies in conducting remote cross-border searches over the Internet.

The emergence of cloud computing poses significant problems for law enforcement agencies. If the WSP engages a cloud computing service provider, its customer data could be stored in data servers around the world. As a result, the law enforcement agent cannot recover any digital evidence from the suspect's computer and from the WSP's servers. Even worse, the WSP would not know the exact physical location of the suspect's data.

Interestingly, our two solutions may work even better in a cloud computing environment. In the first solution, since the exact physical location of the suspect's data is almost impossible to determine, the law enforcement agent has a stronger justification to request the local subsidiary of the WSP to produce the suspect's data regardless of where it is actually stored. In the second solution, the WSP has an urgent need to preserve the suspect's data pending MLA. Because the storage of the suspect's data is outside its control, the WSP has to take steps to preserve the suspect's data immediately or there would be no guarantee that it could produce the suspect's data when the court order arrives.

Countries must address the issue of remote cross-border searches before they encounter increasing numbers of unilateral remote cross-border searches from other countries. If countries with advanced technology do not take the initiative immediately, they may be forced to accept solutions imposed by other countries.

References

[1] P. Bellia, Chasing bits across borders, *University of Chicago Legal Forum*, vol. 2001, pp. 35–101, 2001.

[2] P. Berman, The globalization of jurisdiction, *University of Pennsylvania Law Review*, vol. 151, pp. 311–529, 2002.

[3] S. Brenner and B. Koops, Approaches to cybercrime jurisdiction, *Journal of High Technology Law*, vol. 4(1), 2004.

[4] S. Brenner and J. Schwerha, Transnational evidence gathering and local prosecution of international cybercrime, *John Marshall Journal of Computer and International Law*, vol. 20(3), pp. 347–395, 2002.

[5] M. Brunker, FBI agent charged with hacking, msnbc.com, New York (www.msnbc.msn.com/id/3078784), August 15, 2002.

[6] Council of Europe, Convention on Cybercrime, ETS No. 185, Strasbourg, France (conventions.coe.int/Treaty/EN/Treaties/Html/185 .htm), 2001.

[7] W. Dauterman, Internet regulation: Foreign actors and local harms – At the crossroads of pornography, hate speech and freedom of expression, *North Carolina Journal of International Law and Commercial Regulation*, vol. 28(1), pp. 177–203, 2002.

[8] Financial Times, Schengen at 24, London, United Kingdom (www .ft.com/cms/s/bce06dec-af2f-11dc-880f-0000779fd2ac.html), December 20, 2007.

[9] J. Goldsmith, The Internet and the legitimacy of remote cross-border searches, *University of Chicago Legal Forum*, vol. 2001, pp. 103–118, 2001.

[10] Government of the Hong Kong Special Administrative Region, Computer Crimes Ordinance, Hong Kong (www.infosec.gov.hk /english/ordinances/corresponding.html), 1993.

[11] Government of the Hong Kong Special Administrative Region, Mutual Legal Assistance in Criminal Matters (United States of America) Order, Chapter 525F, Hong Kong (www.legislation.gov.hk/blis_pdf.nsf/6799165D2FEE3FA94825755E0033E532/E002F63F30772ED5482575EF0013F89F?OpenDocument&bt=0), 2000.

[12] W. Graham, Uncovering and eliminating child pornography rings on the Internet: Issues regarding and avenues facilitating law enforcement's access to "Wonderland," *Detroit College of Law at Michigan State University Law Review*, vol. 2, pp. 457–484, 2000.

[13] M. Hirst, *Jurisdiction and the Ambit of the Criminal Law*, Oxford University Press, Oxford, United Kingdom, 2003.

[14] D. Johnson and D. Post, Law and borders: The rise of law in cyberspace, *Stanford Law Review*, vol. 48(5), pp. 1367–1402, 1996.

[15] S. Laville, Legal challenge to web child abuse inquiry, guardian .co.uk, London, United Kingdom (www.guardian.co.uk/uk/2009 /jul/02/web-child-abuse-inquiry-challenge), July 2, 2009.

[16] Supreme Court of Arkansas, Kirwan v. State of Arkansas, *South Western Reporter (Third Series)*, vol. 96, pp. 724–731, 2003.

[17] U.S. Court of Appeals (Eleventh Circuit), United States v. Bank of Nova Scotia, *Federal Reporter (Second Series)*, vol. 740, pp. 817–832, 1984.

[18] U.S. Department of Justice, Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations, Office of Legal Education, Executive Office for United States Attorneys, Washington, DC (www.justice.gov/criminal/cybercrime/ssmanual /ssmanual2009.pdf), 2009.

[19] U.S. District Court (District of Connecticut), United States v. Ivanov, *Federal Supplement (Second Series)*, vol. 175, pp. 367–375, 2001.

[20] U.S. Government, Title 18, Section 2703, *United States Code Annotated, Cumulative Annual Pocket Part*, pp. 87–93, 2009.

[21] E. Wilding, *Computer Evidence: A Forensic Investigation Handbook*, Sweet and Maxwell, London, United Kingdom, 1997.

Chapter 4

AN ANALYSIS OF THE GREEN DAM YOUTH ESCORT SOFTWARE

Frankie Li, Hilton Chan, Kam-Pui Chow and Pierre Lai

Abstract According to official Chinese media sources, the Green Dam Youth Escort (GDYE) software is intended to protect young citizens from viewing unhealthy information on the Internet. However, critics maintain that GDYE has serious security vulnerabilities that allow hackers to take control of computers installed with GDYE. Critics also claim that the software is designed to collect user data and keystrokes for transmission to remote servers for unknown purposes. GDYE was originally mandated to be pre-installed on every computer sold in the People's Republic of China. However, the plan was suddenly shelved in the face of intense international media attention. This paper evaluates the GDYE software's advertised functions and additional non-advertised capabilities. As the software may have spyware and malware functionality, the evaluation monitored the software behavior in a specialized controlled environment. The analysis was performed from a forensics perspective to collect digital evidence and traces in order to prove or disprove that GDYE captures and disseminates private information.

Keywords: Green Dam Youth Escort, analysis, forensic perspective

1. Introduction

Green Dam Youth Escort (GDYE) is an Internet filtering software developed in the People's Republic of China. According to a June 20, 2009 article in *CaiJing Magazine*, GDYE is designed to filter unhealthy information, control surfing time and restrict Internet gaming by Chinese children and youth. With the help of GDYE, parents may view web access logs and screen snapshots.

Under a directive from the Chinese Ministry of Industry and Information Technology (MIIT), GDYE had to be pre-installed on the hard disk and be stored in the recovery partition and on the recovery CD of every personal computer sold in Mainland China on or after July 1, 2009. However, on June 30, 2009, just one day before the deadline, MIIT announced that the mandatory installation of GDYE was postponed to an undetermined date for unspecified reasons.

Starting on July 1, 2009, GDYE was available for download free-of-charge, and was installed on computers in schools, Internet cafes and public locations [8]. According to the China Daily News [4], an MIIT official indicated that the government would definitely carry out its "directive" regarding GDYE, and it was just a matter of time. On August 13, 2009, the Head of MIIT disclosed to the media that GDYE was undergoing a bug fixing process and the plan would be implemented after considering comments from the public. He also disclosed that the option of using a better software system had not been rejected outright.

Since its initial introduction to the public, GDYE has received mixed reactions, both positive and negative, from various entities. One of GDYE's principal goals was to control the access of unhealthy information by Chinese children. However, many individuals are concerned that the government-backed software was created with a hidden agenda to control the flow of information and to restrict the Chinese people from accessing "inappropriate" information on the Internet.

In the light of GDYE's development history and strong government support, it is likely that the software or another similar system will be forced on the public in the near future. However, the questions concerning the leakage of personal information, governmental surveillance and hidden filtering have not been answered. This paper attempts to verify if the software provides special censorship and spying functions that may be used to monitor an individual's online activities.

2. Background

Under an MIIT directive, Zhengzhou Jinhui Computer System Engineering Company (Jinhui) and Beijing Dazheng Human Language Technology Academy (Dazheng) were selected to develop Internet filtering software for MIIT. Project managers from these two companies were subsequently tasked with developing GDYE.

The official websites of Jinhui [7] and Dazheng [5] indicate that the two companies are linked to the Chinese Academy of Sciences, the largest government-funded science and technology research center in China. Jinhui identifies itself as an expert in the area of identifying and filtering pornographic images from the Internet and states that it developed a system that can filter unhealthy images or collect "evidence" from cellular networks. As such, Jinhui was responsible for providing technical expertise related to the image filtering function of GDYE.

Dazheng is a spin-off of the Institute of Acoustics of the Chinese Academy of Sciences. The company website discloses that it invented the notion of a "hierarchical network of concepts" that supports the translation and filtering of Chinese-language messages from computer networks and the Internet. Dazheng is believed to have been responsible for designing and implementing the text filtering function of the GDYE software.

3. Related Work

An analysis of GDYE was released on June 11, 2009 by researchers from the University of Michigan, and an update was published seven days later. The report [10] included the following key findings:

- GDYE contains serious security vulnerabilities due to programming errors that potentially enable websites visited by the user to exploit these problems and seize control of the computer, steal private data, send spam or incorporate the computer in a botnet.
- The GDYE blacklist update process potentially allows the developers, or any third-party impersonating them, to execute malicious code during the filter update.
- The blacklists are taken from CyberSitter and GDYE contains code libraries from OpenCV, an open source image recognition system.

On June 13, 2009, a GDYE update (Version 3.17) was released that supposedly addressed the original web filtering security vulnerability, disabled the blacklists that were copied from CyberSitter, and brought the software into compliance with the OpenCV license. However, a new filtering vulnerability was found. On June 18, 2009, researchers with the Professional Information Security Association (PISA) of Hong Kong demonstrated the re-engineered results of certain binaries of GDYE [11]. Their key findings were:

- The existence of false positive and false negative errors for the URL filtering and pornographic image filtering functions of GDYE.
- GDYE forces certain running processes to close, including Internet Explorer and Microsoft Word and Notepad, when politically-sensitive text (e.g., "June 4 Massacre") is entered.
- Screen snapshots are saved every three minutes by default and the saved information may disclose sensitive information (e.g., details

of online banking sessions, decrypted messages and private communications) to the GDYE administrator.

Faris, Roberts and Wang of the OpenNet Initiative [6] performed a detailed evaluation of the filtering functions by analyzing surfing activities involving Internet Explorer under different versions of GDYE. The goal of the evaluation was "to investigate, expose and analyze Internet filtering and surveillance practices in a credible and non-partisan fashion." Their key findings were:

- GDYE places intrusive controls and actively monitors individual computer behavior by installing components deep in the kernel.
- GDYE provides more functionality than is necessary to protect children online and it subjects users to security risks. GDYE could be used to monitor personal communications and Internet browsing behavior by logging it on the local machine.
- The GDYE implementation for individual computers represents a shift in the filtering strategy to distribute control mechanisms in client-side software in order to offload the burden of sorting through content to individual machines on a network.
- The possibility of personal information leaks could be high (however, the OpenNet Initiative evaluation did not confirm that personal information was being gathered in a central location).

A report by unknown Mainland Chinese researchers [2] indicates that twelve folders and 110 files are created or added to the file system during GDYE installation. After the system is rebooted, four processes and one driver are started and loaded. The process `XNet2.exe` attempts to contact two IP addresses, `211.161.1.134` and `203.171.236.231`, for unknown reasons. Furthermore, the file `XNet2_lang.ini` contains the words "AOption0_1117 = (Upon discovery of harmful information, report automatically to Jinhui Corporation)." This implies that GDYE is capable of sending private information for unknown purposes. GDYE monitors a number of instant messaging applications (e.g., `wow.exe`, `yahoomessenger.exe`, `wangwang.exe` and `qq.exe`) by creating handles using `inject.dll` for Internet Explorer with the clear purpose of collecting private user information. The report further reveals that GDYE monitors TCP and UDP ports to prevent proxy connections if the Free-Gate proxy is used.

Another report [1] discovered that all installation paths of GDYE are contained in the setup file `xstrings.s2g`. After the system is rebooted, it loads `mgtaki.sys` (driver) and starts the execution of services such as `MPSvcC.exe`, `Xnet2.exe` and `XDaemon.exe`. The `XNet2.exe` and `gn.exe` processes are protected to prevent the live deletion of the files. The `kwpwf.dll` file contains the MD5 hash of the admin password, and the magic password `7895123` was found to allow administrative login. The report also noted that the files `cximage.dll`, `CImage.dll`, `xcore.dll`, `Xcv.dll` and `XFImage.xml` are from OpenCV; and the files `HncEng.exe`, `HncEngPS.dll`, `SentenceObj.dll` and `FalunWord.llb` are from Dazheng. Some applications were found to be monitored upon checking the strings information in `injlib.exe`. Finally, when a user surfs the Internet, all content is filtered by WinSock 2 SPI.

Several critics have argued that by applying the blacklist and text filtering functions, GDYE can: (i) filter unwanted political information, including text and images; (ii) act as a tool for the "cyberpolice" to obtain digital evidence of crimes for possible prosecution; and (iii) collect private information from the users, including (but not limited to) web surfing activities and keystrokes, and secretly send the information to certain IP addresses for unknown reasons.

The reviews and reports described above describe the functionality and technical aspects of GDYE. Some of the unconfirmed findings are important, especially the suggestion that GDYE can hide itself in the kernel to open a secret channel that sends private information to certain IP addresses. In view of these and other hidden functions, we treated GDYE as spyware or malware and analyzed it in a controlled environment.

Our work focused on the technical aspects of GDYE and on validating its functionality and behavior while ignoring the political and social rhetoric. In particular, we used digital forensic and reverse engineering tools to scientifically test the hypotheses that (i) using GDYE can result in the loss of private information; and (ii) GDYE is designed to collect private information from users and provide the collected information to centralized servers for unknown purposes.

4. GDYE Analysis

Our objective was to study GDYE's censorship and spying functions. After June 9, 2009, several updates of the GDYE Version 3.17 software were released within a short period of time, supposedly to address the problems pointed out by critics. Due to the presence of multiple GDYE Version 3.17 packages, it is possible that the results presented here may not be reproducible in other packages.

This section describes our analysis of the installation and behavior of GDYE. Installation analysis involved the identification of all the changes

Table 1. MD5 and SHA-1 values for the two GDYE versions.

Version	Function	Hash Value
3.17_000	MD5	`d31aa54dcc339ecdee300c35107f2555`
3.17_000	SHA-1	`4aaa6cec69b4dfd952eda3512a0b45c1f34a0f7c`
3.17_001	MD5	`548c2d2cf32d50a47c69faa8a7640258`
3.17_001	SHA-1	`ee93d0ead4982b53d489b4766d6f96e7618fcd6e`

made to the system during the installation of an official copy of GDYE in the testing environment. Behavioral analysis involved the study of the dynamic behavior of GDYE with emphasis on its supposed censorship and spying functions.

4.1 Installation Analysis

This section describes our GDYE installation procedures and the results of the installation analysis.

GDYE Software Versions GDYE Version 3.17 was the first one analyzed by the public. As mentioned above, a number of problems were reported [10]. Version 3.17 was then removed from the official website and was no longer available for public download. For our evaluation, we obtained the original GDYE Version 3.17 (v. 3.17_000) from a PISA member in Hong Kong. We also evaluated an updated version of GDYE (v. 3.17_001), which we downloaded from the official web site on June 19, 2009.

Upon checking the MAC times and file sizes under file properties, we discovered that v. 3.17_000 with a size of 10,355,637 bytes was modified and accessed on Tuesday, June 9, 2009 at 11:55:10 am. On the other hand, v. 3.17_001 with a size of 10,200,230 bytes was modified and accessed on Saturday, June 13, 2009 at 8:02:38 am. Following standard digital forensic procedures, we calculated the MD5 and SHA-1 hash values for the two GDYE versions (Table 1) and saved them for future reference.

Analysis Environment We set up a specialized, controlled malware analysis environment to test the installation and monitor the behavior of GDYE. The laboratory environment, shown in Figure 1, used the malicious Windows executable (MWC2008) [3]. The Linux version Ubuntu 2.6.28-11-server was used in the Safegate.

The iptables configuration was set to allow DNS to pass through to the Safegate. For Bind9, the query logging functionality was added

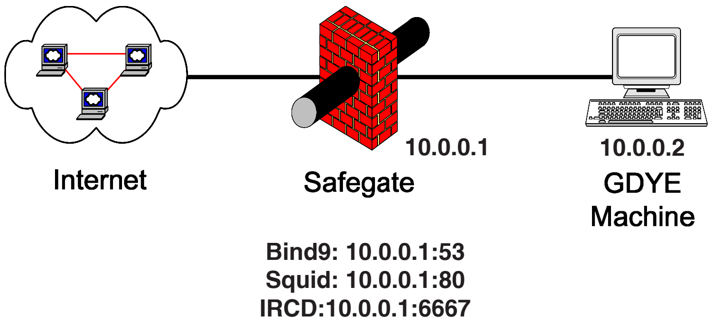


Figure 1. GDYE analysis environment.

to the `named.conf` configuration file. Windows XP SP2 was installed on the GDYE machine without additional patches. Several monitoring and analytical tools, including Autoruns, RegShot and Wireshark, were installed.

Changes During Installation Upon using the RegShot and Autoruns tools, we discovered that the installation process added and modified some Windows registry entries and added files to the %WinDir% folder for XP at C:\Windows and to the %WinSysDir% folder for XP at C:\Windows\system32. By removing some of the prefdata files (i.e., temporary performance-related files created by the Windows system during installation), temporary files and some other unimportant files (e.g., screen files and temporary logs), we found that v. 3.17_000 added 119 files to the system while v. 3.17_001 added 84 files.

An analysis of the Windows registry revealed that both GDYE versions added and modified the same registry keys and values, which allow the automatic loading of the driver (`mgtaki.sys`) and services (`MPSvcC.exe` and `Hnceng.exe`). GDYE also modified the registry key with a filter called `dbfilter.dll` (HKLM\System\CCS\Services\Win Sock2\Parameters\Protocol_Catalog9\Catalog_Entries). This registry key is frequently used by spyware and malware of WinSock hijackers, which is referred to as the Layered Service Provider (LSP) [9]. It is well known that if LSP is not registered properly or if the LSP is buggy, the WinSock catalog in the registry could be corrupted and the computer would no longer be able to access a network. This could be the reason why some GDYE users claim that they lose their Internet connections after unloading the software.

Upon checking the setup file `xstrings.s2g` in the C:\Windows folder with a text editor, we found all the installation paths used by GDYE, which was consistent with the findings in [1]. When we analyzed the bi-

Table 2. Modified files.

Name	Version	Modified Time	Created Time	Size
`adwapp.dat`	3.17_000	4/27/09 6:26:08 am	4/27/09 6:26:08 am	223,572
`adwapp.dat`	3.17_001	6/10/09 1:33:26 am	6/10/09 1:33:26 am	223,674
`dbfilter.dll`	3.17_000	5/22/09 12:47:38 am	5/22/09 12:47:38 am	57,344
`dbfilter.dll`	3.17_001	6/9/09 8:57:06 pm	6/9/09 8:57:06 pm	57,344
`FalunWord.lib`	3.17_000	5/12/09 3:36:14 am	5/12/09 3:36:14 am	5,564,613
`FalunWord.lib`	3.17_001	6/12/09 7:18:36 am	6/12/09 7:18:36 am	5,564,271
`HncStdRun.ini`	3.17_000	7/19/09 10:43:05 am	7/19/09 10:43:05 am	22
`HncStdRun.ini`	3.17_001	8/2/09 7:18:36 am	8/2/09 7:18:36 am	22
`Surfgd.dll`	3.17_000	4/24/09 2:59:36 am	4/24/09 2:59:36 am	126,976
`Surfgd.dll`	3.17_001	6/13/09 6:26:32 am	6/13/09 6:26:32 am	131,072
`XNet2.exe`	3.17_000	5/22/09 5:01:48 am	5/22/09 5:01:48 am	667,648
`XNet2.exe`	3.17_001	6/13/09 7:02:28 am	6/13/09 7:02:28 am	667,648
`xnet2_lang.ini`	3.17_000	7/19/09 11:00:17 am	7/19/09 11:00:17 am	7,748
`xnet2_lang.ini`	3.17_001	8/2/09 7:18:34 am	8/2/09 7:18:34 am	6,842

naries using IDA Pro, we discovered that `XDaemon.exe` and `gn.exe` were not started by the operating system, but by the main process `Xnet2.exe`.

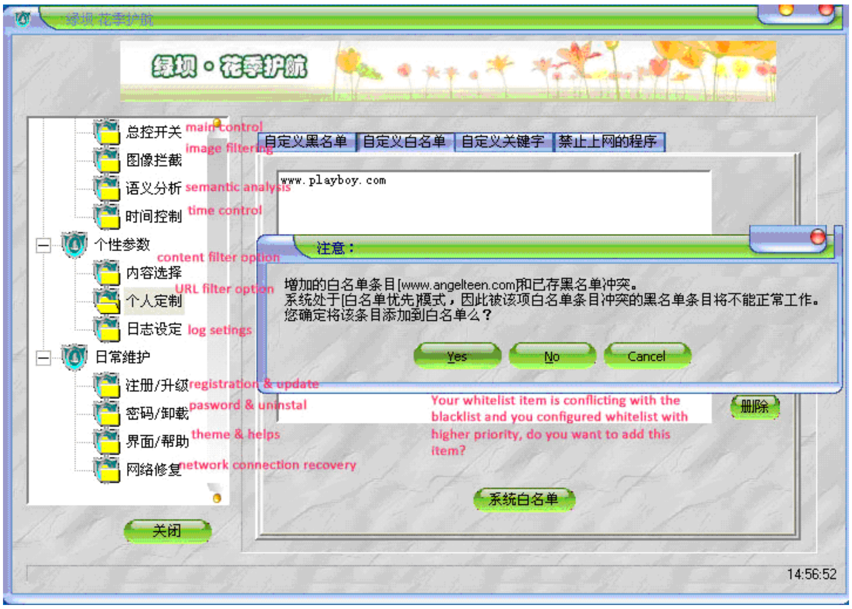
GDYE Version Comparison As mentioned above, a new version of GDYE (v. 3.17_001) was released soon after the first version (v. 3.17_000). The main problems addressed in the new release include the web filtering security vulnerability, the blacklists copied from the CyberSitter program, and the OpenCV license violation. The HashMy-Files tool was used to identify the files that were updated and removed between the two versions. Our analysis revealed that seven files were modified (Table 2) and 35 data files were removed.

Upon checking the modification times of the files, we discovered that the majority of the changes were made between June 9, 2009 and June 13, 2009. The timing of these changes is a strong indicator that the software was modified in response to the public criticism after GDYE's initial release.

To verify the assumption that the removed files were copied from CyberSitter, we conducted an analysis of the data files after applying decoding scripts. We discovered that all the data files that were eliminated in the new version – except for `xwordh.dat`, `xwordl.dat` and `xwordm.dat` – were associated with CyberSitter [10].

4.2 Behavioral Analysis

This section describes our GDYE behavioral analysis procedures and the results of the analysis.



www.playboy.com

增加的白名单条目[www.angelteen.com]和已存黑名单冲突。
系统处于[白名单优先]模式，因此被该项白名单条目冲突的黑名单条目将不能正常工作。
您确定将该条目添加到白名单么？

Your whitelist item is conflicting with the blacklist and you configured whitelist with higher priority, do you want to add this item?

Figure 2. Configuration page.

Filtering GDYE provides three types of filtering functionality: (i) website filtering based on URLs; (ii) image filtering; and (iii) text filtering. It filters most of the popular pornographic and obscene websites (e.g., www.playboy.com, www.sex.com and www.angelteen.com). However, the software administrator can access the configuration page (Figure 2) and add URLs to the whitelist and override the blacklisted URLs. In most cases, the web browser does not display HTML status codes and messages to indicate that a blacklisted URL has been blocked.

GDYE is designed to filter pornographic images. We observed false positive and false negative errors, but no meaningful statistics were obtained with respect to filtering performance. Nevertheless, we discovered that GDYE tends to filter pornographic images that contain light flesh tones rather than dark flesh tones. The software administrator may turn off the image filter function in the configuration page.

GDYE also filtered politically-sensitive phrases such as "June 4 Massacre," "Falun Gong," "Master Li Hongzhi" and "Evil Jiang." When these phrases or others are typed into Microsoft Word or Notepad, GDYE kills the process, which forces the application to crash, causing unsaved work to be lost. We discovered that the time taken for the applications to crash is relatively unpredictable. In some cases, the application does not close immediately after the politically-sensitive phrases are typed, but may close at some point in the future. Note also that the software administrator is unable to access, modify or delete the blacklisted terms.

GDYE also force-closed Internet games such as Warcraft at various stages of game initialization. The software administrator may use the configuration page to restrict Internet connections during specific time periods.

Version Update The software administrator may use the configuration page to manually start the GDYE update process. The DNS query logs indicate that the official GDYE website was accessed during the update process. Wireshark logs showed that the files `kwdata.dat` and `winet.dll` were downloaded.

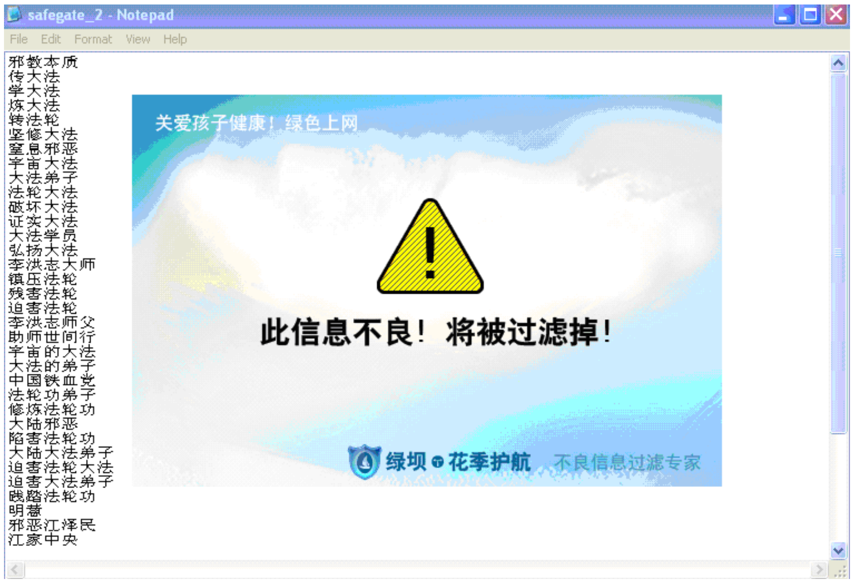
After performing the download, the update process starts automatically and asks for permission to reboot the system. We discovered that the system did not modify or create new keys/values in the Windows registry. Also, several files were modified or removed when updating GDYE v. 3.17_000 to GDYE v. 3.17_001.

Uninstallation An uninstallation option is not provided under the Windows control panel. However, the software administrator may use the configuration page to "unload" the software. Most of the files and folders are removed during uninstallation, except for the driver file `mgtaki.sys` and two folders at C:\Windows\snap and C:\Windows\log. The Internet can be accessed in a normal manner after uninstallation, which indicates that the WinSock SPI registry was handled properly by the uninstallation process.

Spyware and Malware Behavior Additional testing of the malware and logging functionality was performed using Wireshark, FileMon, Regmon and TCPView. The following results were obtained:

- **Snapshots:** GDYE creates screen snapshots in JPEG format every three minutes by default. The snapshot files are saved in the folder C:\Windows\snap. Interestingly, no files are displayed when viewing this folder using Windows Explorer, but issuing a `dir` command under the command prompt will display all the JPEG files. However, if the files are copied to another folder under the command prompt, Windows Explorer can display them without difficulty. All the JPEG files are removed when GDYE is uninstalled.

- **Logging:** Log files are stored in plain text format in the folder C:\Windows\log. As with the snapshots folder, no information is



```
邪教本质
传大法
学大法
炼大法
转法轮
坚修大法
窒息邪恶
宇宙大法
大法弟子
法轮大法
破坏大法
证实大法
大法学员
弘扬大法
李洪志大师
镇压法轮
残害法轮
迫害法轮
李洪志师父
助师世间行
宇宙的大法
大法的弟子
中国铁血党
法轮功弟子
修炼法轮功
大陆邪恶
陷害法轮功
大陆大法弟子
迫害法轮大法
迫害大法弟子
践踏法轮功
明慧
邪恶江泽民
江家中央
```

Figure 3. Standard image displayed when something is filtered.

displayed by Windows Explorer, but the command prompt can be used to copy and display the log files. Also, the files can be viewed using Windows Explorer when they are copied to another folder. However, the log files are removed when GDYE is uninstalled.

- **Pop-Ups:** GDYE does not generate unwanted pop-ups. In some cases, however, GDYE displays the standard image shown in Figure 3 when text is filtered.

- **URL and Text Filtering:** GDYE performs its filtering functions through a WinSock SPI using the `dbfilter.dll` file. It does not display filtered information, but instead displays a normal HTTP message code (402) or performs a TCP reset to the accessing server without sending a message to the browser client. Thus, the user is not notified when an accessed URL has been filtered by GDYE.

- **Connections to External IP Addresses:** Wireshark was used to monitor TCP packets for four 24-hour periods. No Internet activities were seen, except for time synchronization UDP packets sent to the NIST time servers. Also, no obvious Internet transmissions were discovered during the test periods. However, two suspicious IP addresses, `211.161.1.134` and `203.171.236.231`, were found when decoding `XNet2.exe` with IDA Pro. Program messages reading "Preparing to registerK" were found a few bytes after the

IP address `211.161.1.134`, and the message "Report successful" was found two jump blocks after the IP address `203.171.236.231`. Additional tests were performed to check if any actions triggered these two sections of codes, but no network activity related to these IP addresses was discerned in the Wireshark captures.

- **Keystroke Logging:** Upon monitoring the FileMon logs and comparing the changes at different time periods using RegShot, no key logging files were discovered and no files appeared to have been created.

- **Software Uninstallation:** An uninstall option is not provided in the control panel or under the program menu. However, GDYE can be uninstalled from the configuration page by the software administrator. As mentioned above, some folders and files remain in the file system after uninstallation. In addition, one driver was left in the file system.

- **Killing Processes:** Similar to normal malware, the key processes `XNet2.exe`, `XDaemon.exe` and `gn.exe` are protected by handles pointed to each other. When one process is killed, one of the other processes starts up and spawns the killed process as a sub-process.

- **File Modification:** No files were modified in a stealthy manner during our tests.

Vulnerabilities and Exploits Internet Explorer crashed when the system running GDYE was tested against two web filtering vulnerabilities. Two exploit scripts, "Green Dam 3.17 (URL) Remote Buffer Overflow Exploit (XP/SP2)" and "Green Dam Remote Change System Time Exploit," were also tested. The first exploit caused Internet Explorer on the system running GDYE v. 3.17_000 to crash and `calc.exe` to be executed. The second exploit changed the time on the system.

5. Conclusions

Our analysis of GDYE has confirmed most of the concerns about its filtering functions. Our reverse engineering studies indicate that the text filtering function, which is monitored by `injlib32.dll`, force-closes certain applications when blacklisted text is entered; as a result, data loss is unavoidable.

GDYE maintains snapshots and web surfing logs. Any user with the appropriate system permissions can access the information contained in the snapshots and web surfing logs via the command prompt. The web surfing logs are retained in the system folder even after uninstallation.

Also, an individual who knows the magic password can access these logs "legitimately" via the configuration page. While a digital forensic examiner would be delighted to access these logs during an investigation, this aspect of GDYE raises significant privacy concerns. A system running GDYE appears to be vulnerable to custom scripts that inject shell code or exploits, which could incorporate the system in a botnet. Thus, the first hypothesis is proven to be true – GDYE will lead to loss of private information.

However, our tests did not find any evidence of keystroke logging or instances of GDYE transmitting information surreptitiously over the Internet. Although two IP addresses were found in the binary `XNNet2.exe`, no obvious TCP or UDP connections were established to these addresses during our tests. The mere presence of IP addresses in a binary does not necessarily imply that a malicious act was intended. Therefore, our tests do not confirm the critics' concerns that GDYE is designed to collect private information from users and pass it on to centralized servers for unknown purposes. Thus, the second hypothesis must be rejected. Nevertheless, we cannot rule out the possibility that the software will not be modified in the future to implement the collection and forwarding of private information.

References

[1] Anonymous, A technical analysis of the Green Dam Youth Escort software (docs.google.com/View?id=afk7vnz54wt_12f8jzj9gw), 2009.

[2] Anonymous, Green Dam Youth Escort Testing Report (www.meirendaddy.com/blog/?p=404), 2009.

[3] E. Bastuz, Malware Challenge 2008: Behavioral analysis of a malicious Windows executable (www.emre.de/wiki/index.php/MWC 2008), 2008.

[4] J. Cui, X. Wang and X. Cui, Plug not pulled on Green Dam, *China Daily*, Beijing, China (www.chinadaily.com.cn/china/2009-07/02/content_8344967.htm), July 2, 2009.

[5] Dazheng, About Dazheng, Beijing, China (hncit.com/about_us .html).

[6] R. Faris, H. Roberts and S. Wang, China's Green Dam: The implications of government control encroaching on the home PC, Bulletin, OpenNet Initiative, Oxford, United Kingdom (opennet.net/sites/opennet.net/files/GreenDam_bulletin.pdf), 2009.

[7] Jin Hui, About Jin Hui, Zhengzhou, China (www.zzjinhui.com /qyjj.html).

[8] Ministry of Industry and Technology, MITT announcement, Beijing, China (www.miit.gov.cn/n11293472/n11293832/n11293907 /n11368223/12433840.html), June 30, 2009.

[9] H. Wei, J. Ohlund and B. Butterklee, Unraveling the mysteries of writing a WinSock 2 Layered Service Provider, *Microsoft Systems Journal* (www.microsoft.com/msj/0599/LayeredService/LayeredSe rvice.aspx), 2009.

[10] S. Wolchok, R. Yao and J. Halderman, Analysis of the Green Dam censorware system, Revision 2.41, Computer Science and Engineering Division, University of Michigan, Ann Arbor, Michigan (www.cse.umich.edu/∼halderm/pub/gd), 2009.

[11] S. Young, A. Lai, I. Mao, C. Mok, T. Tsang and F. Li, Dissection of Green Dam, presented to the Professional Internet Security Association, Hong Kong, 2009.

II

FORENSIC TECHNIQUES

Chapter 5

FORENSIC ANALYSIS OF A PLAYSTATION 3 CONSOLE

Scott Conrad, Greg Dorn and Philip Craiger

Abstract The Sony PlayStation 3 (PS3) is a powerful gaming console that supports Internet-related activities, local file storage and the playing of Blu-ray movies. The PS3 also allows users to partition and install a secondary operating system on the hard drive. This "desktop-like" functionality along with the encryption of the primary hard drive containing the gaming software raises significant issues related to the forensic analysis of PS3 systems. This paper discusses the PS3 architecture and behavior, and provides recommendations for conducting forensic investigations of PS3 systems.

Keywords: Sony PlayStation 3, gaming console, forensic analysis

1. Introduction

The Sony PlayStation 3 (PS3) hit the Japanese and North American retail markets in November 2006 (March 2007 in Europe) [13]. It is estimated that 75 million consoles will be sold by 2010 [14]. The PS3 marked Sony's entry into the seventh generation of game consoles, which also includes the Nintendo Wii and Microsoft Xbox 360. These gaming consoles possess many of the traits of an Internet-ready home computer; all are designed with internal storage, on-board memory and multimedia capabilities. Furthermore, many of the game consoles can run non-native operating systems (typically Linux-based operating systems), providing them with capabilities beyond those conceived by their manufacturers [8, 10].

Because game consoles provide the same functionality as desktop computers, it should come as no surprise that they have been used in the commission of crimes. An example is the 2009 case of Anthony Scott Oshea of Somerset, Kentucky, who was arrested and charged with possessing child pornography [6]. Investigators discovered that Mr. Oshea's PS3 contained nude pictures of an 11-year-old girl from Houston, Texas; he was eventually convicted of the child pornography charge.

This case and others underscore the need for forensically-sound procedures for the imaging and forensic analysis of (seemingly benign) game consoles with advanced capabilities. Other researchers have focused on the forensic analysis of seventh generation game consoles, including the Xbox [1, 17], Nintendo Wii [16] and PlayStation Portable [3]. The Xbox console has similar functionality as the PS3. However, it lacks the advanced structure and security features of the PS3. Thus, the forensic procedures developed for the Xbox have little, if any, applicability to the PS3.

Another complication is that, unlike desktop computers, game consoles tend to vary greatly in their hardware and software components. This makes it extremely difficult for most forensic examiners to analyze game consoles. Indeed, as of late 2009, no published research exists related to the forensic analysis of the PS3. This paper describes our research on the PS3 system with a focus on developing forensically-sound imaging and analysis procedures.

2. PlayStation 3 Architecture

The PS3 is the most technically advanced system in the seventh generation of gaming consoles [7]. From its initial release in 2006 to 2009, there have been nine different PlayStation models. Each model differs in the configuration of its USB ports, flash card readers, Super-Audio CD support and hard drive size [11]. Each of these differences has potential implications for forensic analysis. The PS3 also incorporates several advanced components, including a Blu-ray disc drive for movies and games, an extremely powerful cell processor (CPU), and an Nvidia RSA graphics processing unit (GPU) [11].

Interestingly, Sony engineers designed the PS3 to allow users to partition the internal hard drive and install a secondary operating system (OS) – typically a distribution of Linux [10] – as long as the OS is capable of supporting the PowerPC architecture. This feature is part of the original design and does not require any modification of the device by the user. In contrast, the Xbox and Wii have to be modified (hacked) in order to install a different OS [5, 15]. Sony's design rationale was that users would desire this feature and providing it would discourage users from modifying the PS3 console in a manner that would release proprietary information or software.

In general, a PS3 console may contain two operating systems: Sony's "Game OS" (native OS) and a second "Other OS" (non-native OS) installed by the user. Note, however, that the PS3 design restricts the size of the Other OS partition to either 10 GB or the difference between the hard drive size and 10 GB [10]; access to certain components is also limited. The ability to install a secondary operating system has been eliminated in the latest (CECH-2000) version of the PS3 [9].

3. Impediments to Forensic Analysis

There are two main impediments to developing forensic procedures for the PS3. First, the PS3 Game OS and file system are proprietary, and it is unlikely that their technical details will be released. Second, security-related measures, such as the mandatory encryption of the Game OS partition, increase the difficulty of recovering evidence.

Some users have modified (hacked) gaming consoles such as the Xbox and Wii [5, 15]; however, these consoles do not use encryption like the PS3. The combination of hard drive encryption and proprietary OS and file system make the task of decrypting a PS3 hard drive problematic, if not impossible. As of late 2009, we know of no case involving the modification of a PS3 system. However, the PS3 does not encrypt the Other OS partition. This means that files located on the non-native OS partition may be identified and recovered.

4. Test Methodology

The forensic tests were performed on an 80 GB CECHK PS3 model purchased from a retail outlet. Several console models were investigated to compare and contrast the results. No modifications of any kind were made to the consoles. The other items used in the tests were a SATA extension cable, an additional 2.5 inch 120 GB hard drive, a USB mouse and a USB keyboard.

The PS3 was connected as stipulated in the instruction manual to an LCD HDTV using an HDMI cable. Internet access was provided via a Category 5 Ethernet cable connected to the Gigabit Ethernet port on the PS3 and attached to the laboratory network. The Other OS was a Ubuntu Linux Desktop v. 8.10. A Knoppix bootable CD was used to zero out the hard drives prior to each test as an experimental control. A Digital Intelligence UltraBlock write blocker was used during imaging.

4.1 Control Boot Test

The first test was conducted to determine if merely turning on (i.e., booting) the PS3 would write to the hard drive. A zeroed hard drive was set up using the PS3; the drive was then removed and imaged. Next, the hard drive was placed back in the PS3 and the console powered up. After approximately three minutes, the console was powered down and the hard drive was removed and imaged. A hex editor was used to compare the two hard drive images.

We found numerous differences between the two images. The differences appeared to occur randomly in block-sized chunks throughout the image. However, they were more numerous towards the beginning of the hard drive and much sparser farther down the drive. Thus, the console writes to the hard drive every time it is powered up. From a forensic standpoint, it is, therefore, important that the hard drive always be removed and imaged using a write blocker. Of course, this is standard operating procedure for traditional laptops and desktop computers.

4.2 Write Blocker Test

This test was conducted to determine if the PS3 software could be executed when a write blocker is placed on the hard drive. The hard drive was removed from the console and placed behind a write blocker. The write blocker was then connected to the console and the system powered up.

We discovered that the PS3 would power up, but not boot up. We turned off the console and removed the write blocker, replacing it with a bridge, the UltraDock Drive Dock (version 4), which would allow writing. The PS3 was then able to power up, boot up and run normally.

The test results suggest that the console must be able to write to the hard drive in order to boot properly (possibly a security measure introduced by Sony). The results also show that the hard drive does not have to be connected directly to the console in order for the PS3 to run.

4.3 Other OS Installation Test

This test was conducted to track the changes that occur to the hard drive when Linux is installed in the Other OS (non-native) partition. A zeroed hard drive was set up using the PS3, and then removed and imaged. Next, the hard drive was inserted back into the PS3 and a 10 GB Other OS partition was created; Linux was immediately installed on this new partition. The hard drive was then removed and imaged, and the two images were compared using a hex editor.

We discovered that the start of the Linux partition is marked by a standard partition table (searching for `0x00000055aa` finds the partition table). Also, as one would expect, the Other OS partition is located at the end of the hard drive. The results suggest that the Linux (or other)

OS is easily located because the Other OS partition and its partition table are not encrypted.

4.4 Imaging over a Network Test

This test was conducted to determine if an unencrypted hard drive image could be obtained. The PS3 was booted into Linux, and `netcat` (a networking service for reading from and writing to network connections) and `dd` (a program for low-level copying and conversion of raw data) were used to create and copy a bit-for-bit image of the hard drive over the network. Next, a second computer (Dell Optiplex 755) was booted using a Knoppix Live CD. The `fdisk` command was used to identify hard disk and partition information.

We discovered that the Linux OS is installed and runs in a partition named `ps3da`. Furthermore, there are only three available partitions (all Linux) with `ps3da` as the boot partition.

Next, `netcat` and `dd` were used to image the hard drive with the PS3 running under Linux. The bits were streamed over the network to the Dell computer. The command used on the PS3 was:

```
dd if=/dev/ps3da | nc [ip address of lab computer] [port number]
```

The command used on the Dell computer was:

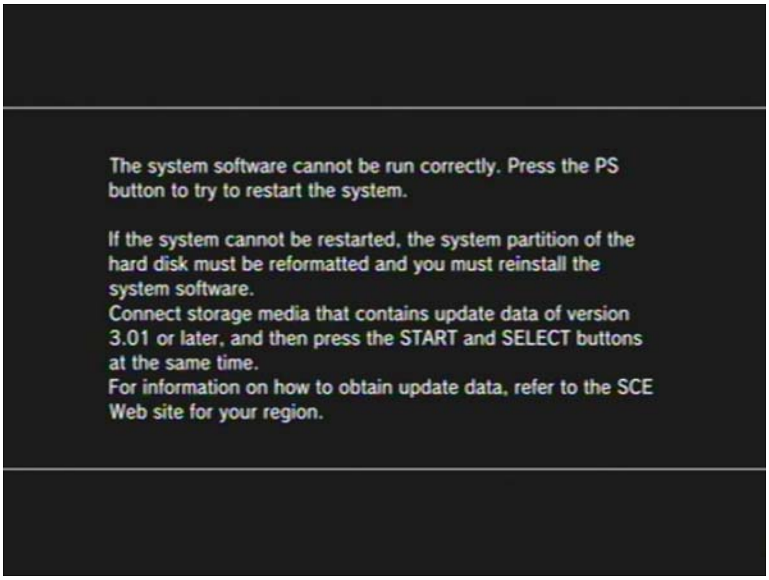
```
nc -l -p [port number] | [image name]
```

The PS3 was powered down after the imaging was completed, and the hard drive was removed and imaged. Comparison of the two images showed that the image obtained over the network is only of the Linux partition and not of the entire drive. The test results suggest that the Game OS partition is inaccessible from an OS running on the Other OS partition.

4.5 Game OS Reinstallation Test

This test was conducted to identify the differences between two PS3 models, CECHK (2008) and CECHE (2007). Changes in the architecture, functionality and behavior of a game console can have significant implications with regard to forensic imaging and analysis. It is, therefore, very important to understand the differences across models.

For this test, the same user account was created on the two PS3s. The hard drives in the two consoles were removed and zeroed, and then replaced in their respective consoles. Upon powering up the systems, we discovered that the newer CECHK model requires the hard drive to be reformatted, the Game OS to be reinstalled and the user account to be recreated (Figure 1). On the other hand, the older CECHE model only requires the hard drive to be reformatted.


The system software cannot be run correctly. Press the PS button to try to restart the system.

If the system cannot be restarted, the system partition of the hard disk must be reformatted and you must reinstall the system software.
Connect storage media that contains update data of version 3.01 or later, and then press the START and SELECT buttons at the same time.
For information on how to obtain update data, refer to the SCE Web site for your region.

Figure 1. Error message to reformat drive.

The results suggest that significant differences in PS3 architecture and behavior may exist between models. Also, the OS and user data are stored in different locations in different models. The newer CECHK model stores data on the hard drive while the older CECHE model stores data in memory on the motherboard.

4.6 Backup Utility Test

This test was conducted to determine what happens when the PS3 backup utility is used to backup the hard drive to a secondary location. The built-in PS3 web browser was used to download several images and bookmark several websites. A second external hard drive was zeroed and formatted with the FAT32 file system. The external hard drive was then connected to the PS3 via a USB port and the PS3 backup utility was used to save the data from the PS3 to the FAT32 hard drive.

Our analysis revealed that the backup has a folder/file structure whose size depends on the amount of data saved on the current hard drive. The name of the folder in the backup is based on the date/time of the backup (e.g., 200910201325). The files contained in the backup are titled "archive" and numbered to differentiate between them; all the files have the `.dat` extension corresponding to data files [4].

The external hard drive was subsequently imaged and examined using AccessData's FTK suite (version 1.80). However, FTK was unable to identify or recover any images from the backup files. Next, a hex editor was used to search for website URLs in the image, but none were located. These results suggest that data cannot be manually recovered from a PS3 backup file (at least for the model tested). It is possible that this is because the backup files are encrypted like the hard drive. Alternatively, the backup files may use a propriety format that only the PS3 can decipher.

4.7 Hard Drive Swap Test 1

This test was performed to determine if the hard drives of two different PS3 consoles (CECHK and CECHE models) can be swapped. A zeroed hard drive was installed in the CECHK console, the system was powered up and the hard drive was formatted using the PS3. The PS3 was then shut down, and the newly formatted hard drive was removed from the CECHK console and installed in the CECHE console. Although the CECHE console did power up, it required the hard drive to be formatted before use.

The test was then reversed. The CECHE console was used to format a zeroed hard drive, which was installed in the CECHK console. Once again, the hard drive had to be formatted before it could be used. The results suggest that a PS3 checks that the hard drive belongs to the specific console before it will boot.

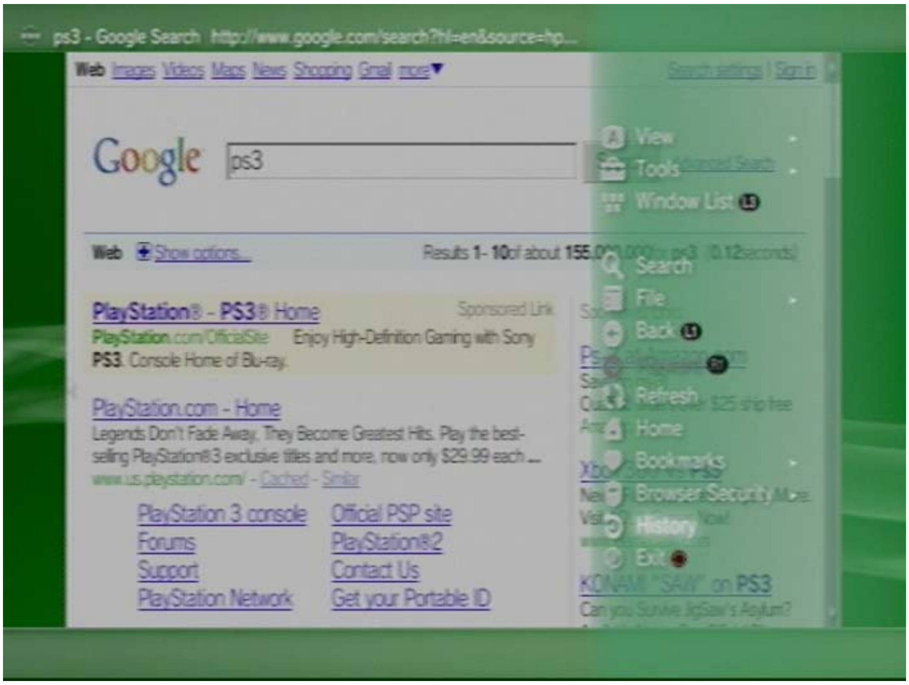
4.8 Hard Drive Swap Test 2

This test was conducted to determine if the hard drives of two different PS3 consoles (CECHK and CECHE models) can be swapped and booted under Linux. A hard drive was zeroed and installed in the CECHK console, the system was powered up and the hard drive was formatted using the PS3. Linux was installed in the Other OS partition of the drive.

The system was restarted to boot under Linux instead of the Game OS. The CECHK console was then powered down and the hard drive removed. Next, the hard drive was installed in the CECHE console and the system was powered up. As in the case of the Hard Drive Swap Test 1, the hard drive had to be formatted before use.

The test was then reversed. The CECHE console was used to format a zeroed hard drive, which was installed and tested in the CECHK console. Once again, the hard drive required formatting before use.

The results suggest that although a PS3 can be set to boot directly into the Linux partition without having to boot into the Game OS partition and that the Linux partition (unlike the Game OS partition) is


ps3 - Google Search http://www.google.com/search?hl=en&source=hp...
Web Images Videos Maps News Shopping Gmail more▼
Search settings | Sign in
Google
ps3
View
Tools
Window List
Search
File
Back
Refresh
Home
Bookmarks
Browser Security Mode
History
Exit

Web Show options... Results 1 - 10 of about 155,000,000 for ps3 (0.12 seconds)

PlayStation® - PS3® Home Sponsored Link
PlayStation.com/OfficialSite Enjoy High-Definition Gaming with Sony PS3. Console Home of Blu-ray.

PlayStation.com - Home
Legends Don't Fade Away. They Become Greatest Hits. Play the best-selling PlayStation®3 exclusive titles and more, now only $29.99 each ...
www.us.playstation.com/ - Cached - Similar

PlayStation 3 console | Official PSP site
Forums | PlayStation®2
Support | Contact Us
PlayStation Network | Get your Portable ID

KONAMI "SAW" on PS3
Can you Survive JigSaw's Asylum?

Figure 2. Accessing the browser history.

not encrypted, the hard drive is nevertheless checked to ensure that it belongs to the specific console before the PS3 will boot.

4.9 Browser Test

This test was performed to determine the number of website URLs that are maintained in the browser history. The browser was used to visit random websites and the browser history was checked to identify the changes after each website was visited (Figure 2).

The results indicate that the browser maintains the last 100 unique websites in its history with the most recently visited URLs at the top of the list (Figure 3). Only one entry is maintained for a site that is visited multiple times; however, the entry moves up the list each time it is accessed. At interesting quirk is that 101 URLs can sometimes be maintained in the browser history. This occurs because each newly-visited URL is added to the history list at the time the website is accessed, but the least recent URL is not deleted until the website has loaded completely. Thus, the browser history can contain 101 URLs if the browser is forced to close before the 101st website is loaded.

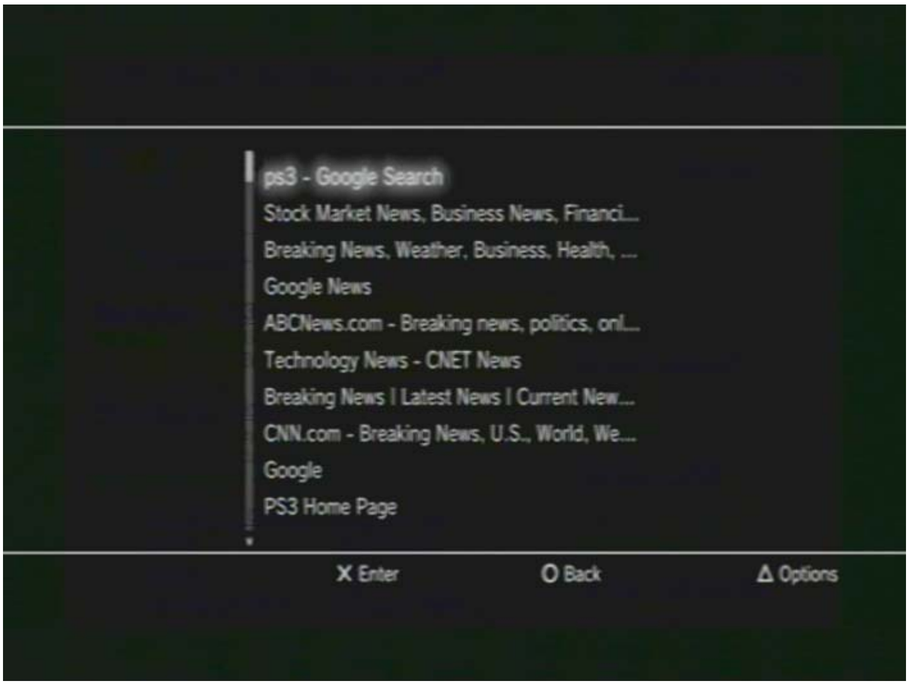

ps3 - Google Search
Stock Market News, Business News, Financi...
Breaking News, Weather, Business, Health, ...
Google News
ABCNews.com - Breaking news, politics, onl...
Technology News - CNET News
Breaking News | Latest News | Current New...
CNN.com - Breaking News, U.S., World, We...
Google
PS3 Home Page

Figure 3. Browser history list.

4.10 Hard Drive Decryption Test

This test was performed to determine if an encrypted PS3 hard drive can be decrypted or read by attaching it externally to the same PS3 that originally encrypted it. Essentially, the goal of the test was to see if a PS3 could read its own hard drive.

The test used two hard drives, one formatted with only the Game OS and the other formatted with a 10 GB Other OS partition containing Linux. The Linux hard drive was inserted into the PS3 and the console was booted into Linux. The second hard drive was then attached to the UltraDock and connected to the PS3 via a USB port. The PS3 running Linux attempted to mount the newly attached drive, but failed every time. The hard drive contents could not be read and, thus, the drive was very likely not automatically decrypted.

5. Evidence Recovery Procedure

The test results suggest that Sony has locked down the PS3 to the point where standard forensic methods do not work. The hard drive is encrypted; the Other OS can be carved out, but the file system cannot be read. The Game OS is completely inaccessible from the Other OS and hard drives can only be read by their respective consoles.

None of the techniques employed were able circumvent the security measures. This does not mean that the PS3 is impenetrable; it is just that the information available at this time is insufficient to defeat the security measures.

Based on the test results, the only means to view encrypted data on a PS3 is to view it natively using the same device. This is the basis of a procedure for obtaining digital evidence from a PS3, which is a reasonable substitute for a traditional forensic method. The evidence recovery procedure, which requires the original PS3 and the hard drive specific to the PS3, involves the following steps:

- Connect a write blocker to the original hard drive. Compute a hash of the hard drive and make a bit-for-bit copy of the drive to a zeroed hard drive of the same size.

- Secure the original hard drive in an evidence locker because it will not be used again. Compute a hash of the copied hard drive in order to check that it is a perfect copy.

- Install the copied hard drive in the PS3.

- Use the PS3 natively to record all settings and to search through the Game OS for files (including in the web browser). Document each step carefully using a video capturing device or application.

This procedure allows for the evidence (original hard drive) to be preserved while allowing its contents to be viewed. The investigation is repeatable because there is no limit to the number of copies that can be made. Capturing the entire procedure on video provides documentation and accountability. This is important because it is inevitable that the copied hard drive will be changed in some manner as demonstrated by the Control Boot Test (Section 4.1), and this cannot be prevented as demonstrated by the Write Blocker Test (Section 4.2).

6. Conclusions

The procedure for obtaining digital evidence from a PS3 serves as a reasonable substitute for a traditional forensic method. Because commercially-available forensic software is unable to read the Game OS and Other OS file systems, additional research is needed to develop a more comprehensive procedure. Traditional methods, such manually carving the data, can only go so far in recovering data from the PS3.

Currently, the PS3 is primarily used for gaming. However, Sony is constantly updating the PS3 firmware to provide new features ranging from enhanced game playing to file sharing and online access [12]. It is certain that attempts will be made to create PS3 viruses and develop attacks that exploit PS3 vulnerabilities. To our knowledge, no viruses exist that target the Game OS and Other OS; however, companies such as Trend Micro have released software to protect PS3s from harmful and inappropriate content [2].

The constant upgrades to the Sony PS3 functionality will only increase the likelihood that these devices will be used in the commission of crimes. Forensic investigators need sophisticated techniques and tools to analyze PS3s and other game consoles. We hope that our research stimulates will renewed efforts in this direction.

References

[1] P. Burke and P. Craiger, Xbox forensics, *Journal of Digital Forensic Practice*, vol. 1(4), pp. 275–282, 2006.

[2] A. Chalk, PlayStation 3 anti-virus software released, *The Escapist*, November 16, 2007.

[3] S. Conrad, C. Rodriguez, C. Marberry and P. Craiger, Forensic analysis of the Sony PlayStation Portable, in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 119–129, 2009.

[4] FileInfo.com, `.dat` file extension (www.fileinfo.com/extension/dat).

[5] A. Huang, Hacking the Xbox (hackingthexbox.com), 2003.

[6] N. Potter, PlayStation sex crime: Criminal used video game to get girl's naked pictures, ABCNews.com, New York (abcnews.go.com /Technology/Story?id=7009977&page=1), March 13, 2009.

[7] Seeking Alpha, Seventh generation gaming consoles: Thinking outside the Box (seekingalpha.com/article/22075-seventh-generation-gaming-consoles-thinking-outside-the-box), December 11, 2006.

[8] Sony Computer Entertainment, Install other OS, Foster City, California (manuals.playstation.net/document/en/ps3/current/set tings/osinstall.html).

[9] Sony Computer Entertainment, New slimmer and lighter PlayStation 3 to hit worldwide market this September, Foster City, California (www.us.playstation.com/News/PressReleases/525), 2009.

[10] Sony Computer Entertainment, Open platform for PlayStation 3, Foster City, California (www.playstation.com/ps3-openplatform /index.html).

[11] Sony Computer Entertainment, PlayStation 3 80 GB system, Foster City, California (www.us.playstation.com/PS3/Systems/Tech Specs/default.html).

[12] Sony Computer Entertainment, PlayStation 3 system software update, Foster City, California (www.us.playstation.com/Support/Sys temUpdates/PS3).

[13] Sony Computer Entertainment, Support: Knowledge Center, Foster City, California (playstation.custhelp.com/app/answers/detail /a_id/232).

[14] T. Surette, Research firm: 75 million PS3s sold by 2010, GameSpot, CBS Interactive, San Francisco, California (www.gamespot.com /news/6163625.html), January 2, 2007.

[15] TechTips, Five things to know before you modify your Wii, Associated Content, New York (www.associatedcontent.com/article/776 395/five_things_to_know_before_you_modify.html?cat=15), May 28, 2008.

[16] B. Turnbull, Forensic investigation of the Nintendo Wii: A first glance, *Small Scale Digital Forensics Journal*, vol. 2(1), pp. 1–7, 2008.

[17] C. Vaughan, Xbox security issues and forensic recovery methodology (utilizing Linux), *Digital Investigation*, vol. 1(3), pp. 165–172, 2004.

Chapter 6

A CONSISTENCY STUDY OF THE WINDOWS REGISTRY

Yuandong Zhu, Joshua James and Pavel Gladyshev

Abstract This paper proposes a novel method for checking the consistency of forensic registry artifacts by gathering event information from the artifacts and analyzing the event sequences based on the associated timestamps. The method helps detect the use of counter-forensic techniques without focusing on one particular counter-forensic tool at a time. Several consistency checking models are presented to verify events derived from registry artifacts. Examples of these models are used to demonstrate how evidence of alteration may be detected.

Keywords: Windows forensics, registry analysis, counter-counter-forensics

1. Introduction

Electronic devices often contain large amounts of data of evidentiary value. However, since it is possible for a suspect to alter the devices through software or hardware, it is extremely important in digital investigations to determine whether or not the evidence collected from seized devices has been modified [5].

This paper advocates the use of information obtained from the Windows registry to verify the consistency of the collected evidence. The Windows registry is a database that stores information about the hardware, software and user profiles of a Windows machine [6]. As such, it an important resource for identifying events that occurred during the use of the machine [1]. During the normal execution of an operating system, certain events always occur in the same order. Therefore, if information about some events is obtained from the registry, then correlating known sequences of these events in temporal order of their timestamps gives a clue as to whether or not the evidence is internally consistent. For example, the event involving the installation of a particular software system for the first time must occur before the event involving the running of the software for the first time. If the timestamp associated with installing the software is later than the timestamp associated with running the software, either the timestamp itself or other information about the event was tampered with.

This paper proposes a method to verify the consistency of registry artifacts by obtaining event information from the artifacts and examining their associated timestamps in event sequences. This provides a generic way to detect the use of counter-forensic techniques without focusing on one particular counter-forensic tool at a time. The method also helps detect when timestamps were altered, which gives a digital forensic investigator additional information about the user's activities.

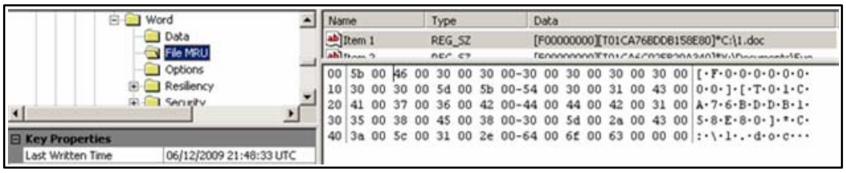
2. Related Research

Previous research on detecting counter-forensic techniques has proceeded along two lines. One approach (see, e.g., [2]) tends to solve the problem from very specific perspective. Experiments are performed to understand the behavior of a particular counter-forensic tool; forensic investigators are then provided with information about where traces of the tool may be found. Although the approach is effective, it suffers from the limitation that the results may not apply to other tools. Indeed, it is almost impossible to develop an individual method against each tool because there are so many counter-forensic tools available [5].

The second, generic approach engages methods that check the consistency of system properties. For example, Gladyshev and Enbacka [3] developed an algorithm to check the consistency of log files. Willassen [7] proposed a technique for discovering evidence of "antedating" by studying the sequence number allocation properties of storage systems. Motivated by these techniques, this paper proposes a general method for verifying the consistency of collected evidence. Although this method may not always be as effective as searching for traces of a counter-forensic tool, it can be used against a wide range of counter-forensic techniques because inconsistencies are detected regardless of the specific technique used to tamper with evidence.

3. Consistency Checking Method

The consistency checking method proposed in this paper involves two steps: (i) obtain events and their timestamp information from the Windows registry; and (ii) verify the consistency of the identified events using the appropriate consistency checking model.



Name	Type	Data
Item 1	REG_SZ	[F00000000][T01CA76BDDB158E80]*C:\1.doc

```
00  5b 00 46 00 30 00 30 00-30 00 30 00 30 00 30 00  [·F·0·0·0·0·0·0·
10  30 00 30 00 5d 00 5b 00-54 00 30 00 31 00 43 00  0·0·]·[·T·0·1·C·
20  41 00 37 00 36 00 42 00-44 00 44 00 42 00 31 00  A·7·6·B·D·D·B·1·
30  35 00 38 00 45 00 38 00-30 00 5d 00 2a 00 43 00  5·8·E·8·0·]·*·C·
40  3a 00 5c 00 31 00 2e 00-64 00 6f 00 63 00 00 00  :·\·1·.·d·o·c···
```

Key Properties	
Last Written Time	06/12/2009 21:48:33 UTC

Extracted Event 1: File MRU key was updated at 06/12/2009 21:48:33 UTC

Extracted Event 2: Value "[F00000000][T01CA76BDDB158E80]*c:\1.doc" was updated into the File MRU key before 06/12/2009 21:48:33 UTC

Figure 1. Event extraction.

3.1 Events and Timestamps

All events obtained from the registry are categorized as extracted events or inferred events. Extracted events are associated with information that was updated at specific registry locations at specific times; these events can be directly extracted from the registry. Inferred events are deduced from the registry contents based on known relationships between registry information and user and system actions.

Extracted Events

An extracted event is either an update of a registry key or a data value corresponding to a key. The time of the update is determined by the LastWrite timestamp of the associated registry key, which specifies the most recent modification time of the key. Figure 1 illustrates the extraction of events from *HKCU\Software\Microsoft\Office\12.0\Word\File MRU*. If the extracted event (Ext Event) corresponds to the updating behavior of the key, then the timestamp can be written as:

$$T^{Ext\ Event(Key)} = T^{LastWrite} \tag{1}$$

Since each data value causes the LastWrite time of the associated key to be updated, the timestamp of the event is estimated by:

$$T^{Ext\ Event(Value\ Data)} \leq T^{LastWrite} \tag{2}$$

Upon combining Equations 1 and 2, the timestamp estimation equation that applies to all possible extracted events is given by:

$$T^{Ext\ Event} \leq T^{LastWrite} \tag{3}$$

A better way to estimate the timestamp for an extracted event associated with a particular record is to use the registry snapshot comparison method [9]. This method helps bind an extracted event to a specific time interval. The comparison method is based on the fact that many Microsoft Windows versions automatically back up system registry hives, creating "system restore points" approximately every 24 hours. By considering each registry snapshot as a previous state of the system registry and comparing a given state of the registry with previous registry snapshots, changes made to the registry between the creation times of two consecutive restore points can be identified. When previous registry snapshots are available, comparing each registry snapshot with its preceding snapshot can identify extracted events that are known to have occurred before the LastWrite timestamp of the key and also after the creation time of the preceding registry snapshot:

$$T^{Prec\ Snapshot} \leq T^{Identified\ Ext\ Event} \leq T^{LastWrite} \tag{4}$$

Inferred Events

Carvey [1] has described the inference of user and system events from the registry. In general, the time interval between an inferred event and the action that triggered the corresponding update in the registry (i.e., the corresponding extracted event) influences the selection of the consistency checking model that is applied. Thus, inferred events are divided into three groups:

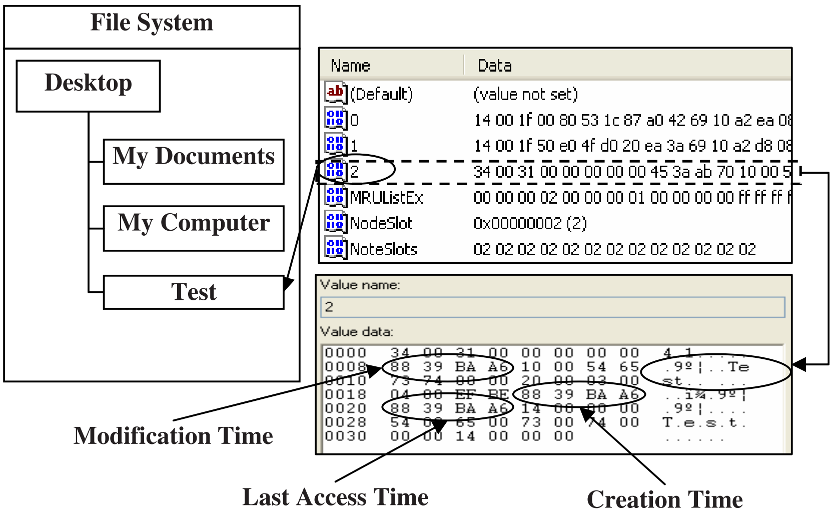
- Inferred events that occurred before the corresponding extracted event.
- Inferred events that occurred "at the same time" as the corresponding extracted event.
- Inferred events that occurred after the corresponding extracted event.

Note that an inferred event and its associated extracted event do not occur simultaneously. However, if the time interval is short enough, the extracted event can be assumed to have occurred "at the same time" as the action that precipitated it.

In the following, we present four examples of inferred events to demonstrate how events may be inferred from registry information.

The first example involves ShellBag information associated with the "Test" folder on a local computer (Figure 2). As described in [8], the registry value associated with this folder includes the timestamp when the ShellBag information was first updated in the registry. Therefore, three events can be inferred:

- Folder "Test" located on the Desktop was created at `08/12/2008 20:53:52`.



Name	Data
(Default)	(value not set)
0	14 00 1f 00 80 53 1c 87 a0 42 69 10 a2 ea 08
1	14 00 1f 50 e0 4f d0 20 ea 3a 69 10 a2 d8 08
2	34 00 31 00 00 00 00 00 45 3a ab 70 10 00 5
MRUListEx	00 00 00 02 00 00 00 01 00 00 00 00 ff ff ff f
NodeSlot	0x00000002 (2)
NoteSlots	02 02 02 02 02 02 02 02 02 02 02 02 02

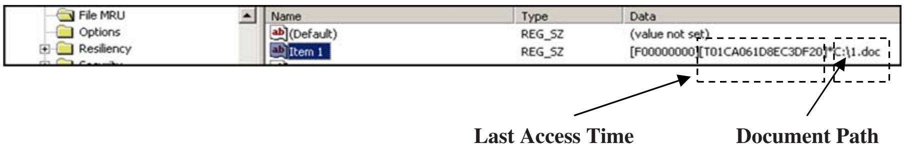
Value name:
2

Value data:
```
0000   34  00  31  00  00  00  00  00   4.1.....
0008   88  39  BA  A6  10  00  54  65   .9º¦..Te
0010   73  74  00  00  20  00  03  00   st......
0018   04  00  EF  BE  88  39  BA  A6   ..ï¾.9º¦
0020   88  39  BA  A6  14  00  00  00   .9º¦....
0028   54  00  65  00  73  00  74  00   T.e.s.t.
0030   00  00  14  00  00  00           ......
```

Figure 2. ShellBag information associated with the "Test" folder.

- Folder "Test" located on the Desktop was accessed at `08/12/2008 20:53:52`.
- Folder "Test" located on the Desktop was modified at `08/12/2008 20:53:52`.

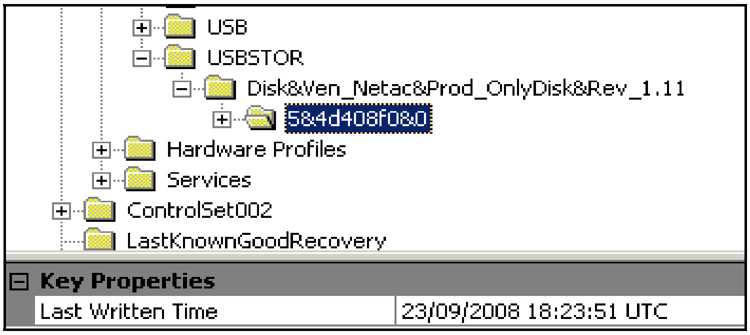
In this example, more than one event is inferred from a single registry record. Since the consistency checking method is based on the events and timestamps that are found, it is important to infer as many events as possible from the registry. Note that all three events are known to have occurred before the corresponding extracted event.



Name	Type	Data
(Default)	REG_SZ	(value not set)
Item 1	REG_SZ	[F00000000][T01CA061D8EC3DF20]*C:\1.doc

Figure 3. FileMRU key.

The second example involves the FileMRU key, which is located at *HKCU\Software\Office\12.0\Microsoft\Word\FileMRU* (Figure 3). It stores the paths of Word documents that have recently been opened by Microsoft Office. The information in Figure 3 implies the (inferred) event that a Word document *C:\1.doc* was accessed at `13:58:55 16/07/2009` corresponding to the Windows 64-bit timestamp `01CA061D8EC3DF20`.



Last Written Time 23/09/2008 18:23:51 UTC

Figure 4. USBSTOR subkey.

The third example involves the USBSTOR subkey located at *HKLM\System\ControlSet01\Enum\USBSTOR* (Figure 4). The inferred event is that a USB device Disk&Ven_Netac&Pro_OnlyDisk&Rev_1.11 was connected to the system at `23/09/2008 18:23:51`. The timestamp is estimated using the extracted event: Registry key *HKLM\System\ControlSet-01\Enum\USBSTOR\Disk&Ven_Netac&Pro_OnlyDisk&Rev_1.11\5&4d408f08&0* was updated at `23/09/2008 18:23:51`. The timestamp of the extracted event can be used for this inferred event because the inferred event belongs to the second type of inferred event (defined above), where the extracted event is assumed to have occurred at the same time as the inferred event.

The fourth example corresponds to the third type of inferred event that occurred after the corresponding extracted event. The value Lease TerminatesTime under the registry key *HKLM\SYSTEM\ControlSet001\Services\Tcpip\Parameters\Interfaces\AdapterID* gives the time that the IP address of the adapter expired. This definitely occurred after the adaptor was connected.

3.2 Consistency Checking Models

After obtaining event information from the registry, it is necessary to verify the events and their associated timestamp information. This section describes several consistency models that may be used for this purpose. Because each event can be placed in a different position in a time sequence when grouped with other events, it is difficult, if not impossible, to define a consistency checking model for an event without considering the details of other events. We address this issue in a generic manner using a context-based model. Each model is defined with respect to a specific context; the model is applied when the events meet the associated conditions.

Basic Model

The basic event-time bounding model is used to estimate the time frame during which a particular event without time information occurred. This is done by considering its relation in time to other events that are known to have occurred either before or after the event [4]. The same concept can also be applied to user data. Since this model also represents the relationships between a sequence of multiple events and their timestamps, it is the basis of other consistency checking models. The event-time bounding model utilizes two rules:

- **Rule 1:** If Event A occurred before Event B, and Event B occurred before Event C, then the time that Event B occurred is bounded by the times that Events A and C occurred:

$$T^A < T^B < T^C \tag{5}$$

- **Rule 2:** If several Events A_1, A_2, ..., A_m occurred before Event B, and Event B occurred before several Events C_1, C_2, ..., C_n, then the time that Event B occurred is bounded by:

$$Max(T^{A_1},\ T^{A_2}, \ldots, T^{A_m}) < T^B < Min(T^{C_1},\ T^{C_2}, \ldots, T^{Cn}) \tag{6}$$

Checking Inferred Events and Extracted Events

The basic model was developed to verify the consistency of inferred events and extracted events. As mentioned earlier, there are three types of inferred events. Therefore, based on their relationship with the corresponding extracted event, three equations can be derived using Equations 3 and 5.

- For an inferred event (Inf Event) that occurred before the extracted event, we have:

$$T^{Inf\ Event} < T^{Ext\ Event} \leq T^{LastWrite}$$

- For an inferred event that occurred at the same time as the extracted event, we have:

$$T^{Inf\ Event} = T^{Ext\ Event} \leq T^{LastWrite}$$

- For an inferred event that occurred after the extracted event, we have:

$$T^{Inf\ Event} > T^{Ext\ Event}$$

or, if the time difference Δ between the action and a future action is known, we have:

$$T^{Inf\ Event} = T^{Ext\ Event} + \Delta$$

In the example in Figure 2, if the LastWrite timestamp of the key is `07/12/2008 11:00:00`, then based on the model for inferred events that occurred before the corresponding extracted event, the information conflicts with the inferred event: Folder "Test" was created at `08/12/2008 20:53:52`.

Additionally, if multiple inferred events are identified as being associated with the same extracted event, the timestamp of the extracted event will be affected by all the associated inferred events. Therefore, the consistency checking models must be extended according to Equations 3 and 6.

- For inferred events that occurred before the extracted event, we have:
$$Max(T^{Inf\ Events}) < T^{Ext\ Event} \leq T^{LastWrite}$$

- For inferred events that occurred at the same time as the extracted event, we have:
$$Max(T^{Inf\ Events}) = T^{Ext\ Event} \leq T^{LastWrite}$$

- For inferred events that occurred after the extracted event, we have:
$$Min(T^{Inf\ Events}) > T^{Ext\ Event}$$

The application of the model is illustrated using the UserAssist key at *HKCU\Software\Windows\CurrentVersion\Explorer\UserAssist.* This key stores a list of values that record information about the execution of software on the computer. Each time a particular software executes, the corresponding value is updated under the UserAssist key. This falls under the model for inferred events that occurred at the same time as the corresponding extracted event. Assume that three values are found under the UserAssist key as shown in Table 1. Upon applying the model, Max($T^{Inf\ Events}$) is equal to `15:35:12 14/05/2009`. If the LastWrite timestamp is `20:04:31 25/04/2009`, then the model proves that the event information is inconsistent.

Table 1. Interpreted information for the UserAssist key.

Software Path	**Timestamp**
C:\Program Files\Internet Explorer\iexplorer.exe	`10:03:01 05/03/2009`
C:\WINDOWS\system32\notepad.exe	`15:35:12 14/05/2009`
C:\Program Files\Windows Media Player\wmplayer.exe	`19:11:52 22/01/2009`

Checking Inferred Events

When examining the registry, it is possible to infer events from different locations in the registry that point to the same user or system action. This is because information is sometimes saved at multiple locations in the registry. The consistency of inferred events can be verified using these related pieces of data.

Zhu, *et al.* [8] have proposed several rules that use ShellBag information to determine if a folder was accessed. Specifically, the information under the BagMRU registry key (located at *HKCU\Software\Microsoft\Windows\ShellNoRoam\BagMRU*) and the Bags key (located at *HKCU\Software\Microsoft\Windows\ShellNoRoam\Bags*) can be used to determine if the folder was accessed during a particular period. If one event, i.e., the folder was accessed at `10/07/2009 7:30:00` is inferred from information in the Bags key, and another event, i.e., the same folder was not accessed between `09/07/2009 9:12:00` and `11/07/2009 18:20:30` is inferred from the BagMRU key, then it follows that the two inferred events are not consistent.

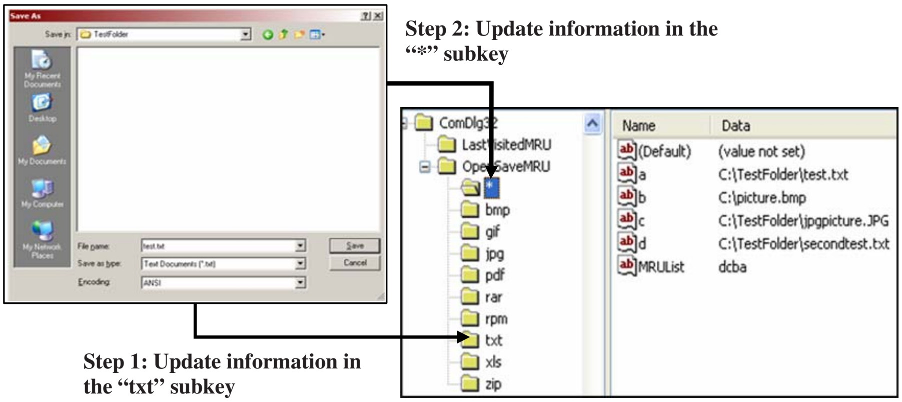
Checking Extracted Events

The consistency between extracted events is due to the fact that some registry keys are always updated after other keys. When events pertaining to registry operation are extracted, their timestamps should appear in the same order. By applying the basic model to this relationship it is possible to say that: if Key B is always updated after Key A, then the most recent extracted event of Key B is definitely after the most recent extracted event of Key A:

$$Max(T^{Ext\ Events(Key\ A)}) < Max(T^{Ext\ Events(Key\ B)})$$

Because the most recent extracted event of a registry key is equal to the LastWrite timestamp of the key, the consistency check can be expressed as:

$$T^{LastWrite(KeyA)} < T^{LastWrite(KeyB)}$$



Name	Data
(Default)	(value not set)
a	C:\TestFolder\test.txt
b	C:\picture.bmp
c	C:\TestFolder\jpgpicture.JPG
d	C:\TestFolder\secondtest.txt
MRUList	dcba

Figure 5. OpenSaveMRU update example.

Similarly, if Key B is always updated after other Keys A_1, A_2, ..., A_n, then the timestamp of Key B is constrained by the most recent extracted events:

$$Max(Max(T^{Ext\ Events(Key\ A_1)}), \ldots, Max(T^{Ext\ Events(Key\ A_n)})) \\ < Max(T^{Ext\ Events(Key\ B)})$$

Upon replacing the most recent extracted event with the LastWrite timestamp of the key, the consistency check can be written as:

$$Max(T^{LastWrite(Key\ A_1)}, \ldots, T^{LastWrite(Key\ A_n)}) < T^{LastWrite(Key\ B)}$$

This model can be used to check the consistency of event information stored in the OpenSaveMRU key at *HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU*. The key is known to record the most recently used history of the open and save dialog of the Windows system. For example, downloading a file and using the open and save window to save the file in a local directory updates this key with information about the selected directory.

Figure 5 presents the structure of the OpenSaveMRU key. Subkeys in OpenSaveMRU correspond to files with extensions (e.g., "jpg" corresponding to a `.jpg` file and "rar" corresponding to a compressed `.rar` file). The OpenSaveMRU key itself saves information about files without extensions. The only exception is that the subkey "*" records directories for every file.

The algorithm used to update the OpenSaveMRU key and its subkeys checks if the open and save window is used. If it is, the path of the file is stored in the "*" subkey after updating the other file extension

Table 2. Interpreted information of OpenSaveMRU.

Registry Key Path	Timestamp
OpenSaveMRU\bmp	07:42:16 13/07/2009
OpenSaveMRU\jpg	22:12:28 21/07/2009
OpenSaveMRU\txt	21:15:42 09/07/2009
OpenSaveMRU\*	19:27:09 14/07/2009

type keys. In the example in Figure 5, the file *C:\TestFolder\test.txt* was first stored in the "txt" subkey, and then saved in the "*" subkey. That is to say, the "*" key is always updated after other OpenSaveMRU keys or subkeys. Thus, the "*" key satisfies the conditions described above. Table 2 presents the timestamps of all the keys; the information is inconsistent because the timestamp of the OpenSaveMRU\txt key is later than the timestamp of the OpenSaveMRU* key.

Checking Registry Events and Other Events

Apart from the registry database, there are other sources that provide information about events that have occurred. Two important sources of events that should be checked for consistency are file timestamps and registry snapshots.

- **File Timestamps:** In most versions of Microsoft Windows, three timestamps are associated with a file, corresponding to when the file was last accessed, last modified and created. Each extracted event obtained from the registry indicates an update of the registry file, so the extracted event should cause the modification timestamp of the registry file to be updated if the file handle is properly closed after the update. The consistency check for this situation is:

$$Max(T^{Ext\ Event\ 1}, \ldots, T^{Ext\ Event\ n}) \leq T^{Reg\ Mod\ Time}$$

  A better method to implement this check is to compare the LastWrite time of each extracted event with the last modification time of each file because Max($T^{ExtEvents}$) is equal to $T^{LastWriteKey}$. The consistency check for this situation is:

$$Max(T^{LastWrite\ 1}, \ldots, T^{LastWrite\ n}) \leq T^{Reg\ Mod\ Time}$$

  Note that this inequality does not apply when the registry file handle is not closed properly due to abnormal termination.

- **Registry Snapshots:** Each registry snapshot is created by updating the contents of the preceding registry snapshot. Therefore, any events that have occurred between two snapshots will appear after the creation time of the preceding snapshot:

$$T^{Prec\ Snapshot} < Min(T^{Ext\ Event\ 1}, \ldots, T^{Ext\ Event\ n})$$

$$T^{Prec\ Snapshot} < Min(T^{Inf\ Event\ 1}, \ldots, T^{Inf\ Event\ n})$$

  Whether an extracted event has occurred or not is determined by comparing the timestamps of a registry key in the snapshot and in the next snapshot. If the timestamp is updated in the newest snapshot, then it implies that the corresponding extracted event did occur. According to Equation 3, the timestamp of the extracted event cannot be after the updated LastWrite timestamp. Therefore, the consistency check for this situation is:

$$T^{Prec\ Snapshot} \leq Min(T^{Updated\ LastWrite\ 1}, \ldots, T^{Updated\ LastWrite\ n})$$

3.3 Other Consistency Checking Considerations

After identifying inconsistent information, there may be a need to understand how the inconsistency was created. As mentioned above, if the system runs without any intentional changes, it will be consistent all the time. Some of the reasons for inconsistent information are:

- **System Clock Adjustment:** Many timestamps are based on the current system clock, especially the LastWrite timestamp associated with each key. If the system clock is temporarily adjusted, it may leave detectible inconsistencies in the timestamps recorded during the altered time period. A consistency check may identify these inconsistencies if they are not overwritten by new information.

- **Registry Information Modification:** It is often the case that the registry API is used to modify registry information. While any part of the registry can be modified, the LastWrite timestamp of the key is also updated because the system considers the invocation of the API as a normal registry operation. Therefore, it is possible to identify this trace by examining the consistency of extracted events. An inconsistent timestamp may imply that tampering has occurred. For example, in Table 2, if RegEdit was used to modify the contents of the `jpg` subkey at `22:12:28 21/07/2009`, the inconsistency shown in Table 2 would be produced.

- **Registry Hive Modification:** A file editor tool can be used to modify the registry hive directly. This is hard to implement in practice because it requires the user to understand the unique structure of the particular registry hive. Also, it is difficult to detect if the registry has been modified in this manner because the registry is not automatically updated when the modification is made unless the timestamp is also edited at the same time.

4. Conclusions

The method for checking the consistency of registry information involves extracting and inferring events and their corresponding timestamps from the registry database. Appropriate consistency checking models are then used to verify the information that is collected and help detect counter-forensic activity. Our future research will examine the potential of using other registry information as well as data from other sources in sophisticated consistency models.

Acknowledgements

This research was funded by the Science Foundation of Ireland under the Research Frontiers Program 2007 Grant No. CMSF575.

References

[1] H. Carvey, *Windows Forensic Analysis*, Syngress, Burlington, Massachusetts, 2007.

[2] M. Geiger and F. Cranor, Counter-Forensic Privacy Tools: A Forensic Evaluation, Technical Report CMU-ISRI-05-119, Institute for Software Research International, Carnegie-Mellon University, Pittsburgh, Pennsylvania (reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-119.pdf), 2005.

[3] P. Gladyshev and A. Enbacka, Rigorous development of automated inconsistency checks for digital evidence using the B method, *International Journal of Digital Evidence*, vol. 6(2), pp. 1–21, 2007.

[4] P. Gladyshev and A. Patel, Formalizing event time bounding in digital investigations, *International Journal of Digital Evidence*, vol. 4(2), pp. 1–14, 2005.

[5] S. Hilley, Anti-forensics with a small army of exploits, *Digital Investigation*, vol. 4(1), pp. 13–15, 2007.

[6] Microsoft Corporation, Windows registry information for advanced users, Redmond, Washington (support.microsoft.com/kb/256986), 2008.

[7] S. Willassen, Hypothesis-based investigation of digital timestamps, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 75–86, 2008.

[8] Y. Zhu, P. Gladyshev and J. James, Using ShellBag information to reconstruct user activities, *Digital Investigation*, vol. 6(S1), pp. S69–S77, 2009.

[9] Y. Zhu, J. James and P. Gladyshev, A comparative methodology for the reconstruction of digital events using Windows restore points, *Digital Investigation*, vol. 6(1-2), pp. 8–15, 2009.

Chapter 7

FORENSIC TRACKING AND MOBILITY PREDICTION IN VEHICULAR NETWORKS

Saif Al-Kuwari and Stephen Wolthusen

Abstract Vehicular networks have attracted significant attention in the context of traffic control and management applications. However, vehicular networks also have important applications in crime prevention and criminal investigations. This paper presents a system for passively tracking a target vehicle whose driver is assumed to be a "person of interest." The tracking system relies on the dynamic recruitment of neighboring vehicles of the target as agents. A mobility prediction algorithm is used to probabilistically predict the target's future movement and to adjust the tracking process. Combining agent-based tracking and mobility prediction enables a target vehicle to be passively localized and tracked in an efficient manner.

Keywords: Vehicular networks, passive tracking, mobility prediction

1. Introduction

Due to the heightened interest in driver safety and infotainment applications, vehicular networks along with other mature wireless technologies will be widely deployed in future automobiles. This has motivated the research community to investigate various aspects of vehicular networks.

The main motive behind the emergence of vehicular networks is driver safety. However, these networks have important applications in crime prevention and criminal investigations in which law enforcement agencies must track the movements of "persons of interest."

Vehicular network algorithms differ from those encountered in conventional mobile networks. Vehicular network nodes move more rapidly than the nodes in other mobile networks. However, vehicular nodes are characterized by somewhat limited freedom of motion since their movements are constrained by roadways and traffic regulations. This introduces additional challenges for vehicular tracking algorithms because they must efficiently adapt to the peculiarities of vehicular movements; but this also improves their mobility predictions because nodes move along predefined paths.

This paper presents a passive vehicular tracking system that relies on the dynamic recruitment of neighboring vehicles as agents. A mobility prediction algorithm is used to probabilistically predict the target's future movement and to adjust the tracking process. Combining agent-based tracking and mobility prediction enables a target vehicle to be localized and tracked both efficiently and clandestinely.

2. Related Work

Boukerche, *et al.* [3] discuss localization techniques that can be used in vehicular networks along with their practical implications. The authors demonstrate that most localization techniques suffer from inherent inaccuracies that may not be acceptable for vehicular-based applications that require precise location information. In such situations, the best solution is to use data fusion where the results of multiple localization techniques are combined to increase accuracy.

While localization techniques for vehicular networks are usually GPS-based, not all vehicles are equipped with GPS devices. Also, GPS-based techniques are useless when GPS signals are not available (e.g., inside tunnels). Benslimane [2] addresses this problem in an extension to the ODAM messaging dissemination protocol, enabling vehicles that are not GPS-capable to be localized.

A popular tracking technique that is well suited to vehicular scenarios involves map-matching tracking, which attempts to match a node's actual location (raw) data to maps. Barakatsoulas, *et al.* [4] present several such algorithms that exploit vehicular trajectory information. However, this approach is not suitable for applications that require real-time tracking. Other tracking techniques such as installing tracking tags on a target vehicle [9] also exist, but these solutions are not ad hoc and require additional preparation and overhead.

3. Vehicular Networks

Generally, vehicular networks are based on ad hoc infrastructures and are, therefore, referred to as vehicular ad hoc networks (VANET), a subclass of mobile ad hoc networks (MANETs). The U.S. Federal Communications Commission (FCC) has allocated a 75 MHz spectrum in the 5.9 GHz band (5.850 GHz to 5.925 GHz) for vehicular communications, which can either be vehicle to vehicle (V2V) or vehicle to infrastructure (V2I) communications. V2V depends on an ad hoc infrastructure where vehicles directly exchange information like accident and congestion warnings. V2I, on the other hand, assumes the presence of pre-installed road components that vehicles communicate with in order to retrieve information. However, the cost of installing these components often limits V2I applications.

Communication in vehicular networks is based on the conventional IEEE 802.11 wireless standard (WiFi) or 3G via CDMA (code division multiple access) technology. WiFi is simpler and less expensive to deploy, but has lower reliability because it was not designed to operate in environments with rapid movements. In contrast, 3G is more relevant to vehicular communications, but is more expensive and difficult to deploy because of its centralized architecture [8]. In this paper, we assume that vehicles communicate using WiFi, which is a more widely deployed solution. Note that WiFi has some efficiency limitations, especially because it operates in the half-duplex mode (i.e., two nodes cannot communicate with each other simultaneously).

4. Mobility in Vehicular Networks

Mobility models represent the motion behavior of vehicles. Mobility models designed to simulate the movement of nodes in MANETs can be used to simulate vehicular movements in VANETs and most vehicular simulation approaches employ these models. However, MANET models have proved unreliable and unrealistic, which has motivated the design and development of mobility models targeted for vehicular networks.

Vehicular mobility models are categorized as microscopic, mesoscopic and macroscopic models. Microscopic models describe the dynamics of the individual movement of each vehicle and its interaction with other vehicles (this may require excessive computational resources for large-scale simulations). Macroscopic models simulate the characteristics of the roadways (motion constraints) and the flow of vehicles, but do not consider individual vehicle movement, which impacts their degree of realism. Interested readers are referred to [1] for a discussion of macrosopic models versus microscopic models. Mesoscopic models belong to a new "hybrid" class of vehicular mobility models that combine features of microscopic and macroscopic models. In this paper, we adopt the microscopic intelligent driver motion (IDM) model [10] along with two extensions, IDM-IM (IDM with intersection management) and IDM-LC (IDM with lane changes).

5. Vehicle Localization

Conventional localization techniques used in cellular and sensor networks can also be applied to vehicular networks, albeit with slightly degraded accuracy. These techniques involve measuring the received signal strength (RSS), time of arrival (TOA) or time difference of arrival (TDOA). Since we adopt a passive tracking approach, only RSS is relevant because, unlike TOA and TDOA, it does not require a fully synchronized network and active communication with the target.

The RSS technique involves measuring the strength of the signals emitted from the target vehicle in order to estimate the distance to the target. RSS measurements are nonlinear due to radio propagation effects; nevertheless, our algorithm adopts a free space radio propagation model, which assumes a direct line of sight path between vehicles. This assumption does not severely degrade the accuracy of the localization process because it is most likely that the only obstacles would be the moving vehicles and their effects can be eliminated by averaging the measurements. Note that vehicular acceleration also affects RSS measurements, but we assume that this effect can be mitigated by adding noise in the computations.

The localization of a target vehicle C_s involves three of its neighboring vehicles. However, only two of these neighbors (tracking agents) track the target; the other (localizing agent) is retired immediately after the localization is completed. We denote the tracking agents as C_a (main) and C_b (backup) and the localizing agent as C_l. We assume that the three agents can localize themselves using GPS, which is fairly common in current vehicles.

The localization algorithm has four steps:

- **Step 1 (Recruitment):** The tracking agents are recruited by randomly requesting RSS readings from nodes around the target. The nodes with the highest readings are recruited. Details about the initial and subsequent recruitment process are provided in Section 7.

- **Step 2 (Distance Measurement):** The distances between the recruited agents and the target are estimated using the Friis equation [6]:

$$d = \sqrt{P_t A_r / 4\pi P_r}$$

  where P_r and P_t are the received and transmitted power, respectively; and A_r is the effective area of the receiving antenna.

- **Step 3 (Trilateration):** The location of the target is estimated using a geometric transformation to find the intersection point of the three circles centered at the agents that have radii equal to the respective distances of the agents from the target as measured in Step 2.

- **Step 4 (Retirement):** The localizing agent C_l is retired.

The localization algorithm can also be used in conjunction with the prediction algorithm (Section 6) to determine when the next recruitment is required. This assumes that the agent can measure its own speed, which is possible because it is equipped with a GPS device. The average speed is estimated by dividing a particular distance by its corresponding traversal time.

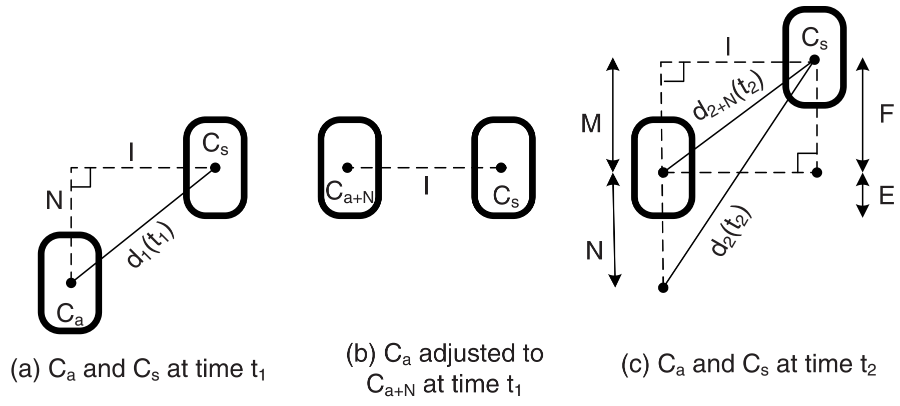
6. Mobility Prediction

Mobility prediction is an important aspect to consider when tracking vehicles. Generally, mobility prediction in vehicular networks is slightly simpler in VANETs than MANETs because vehicular mobility is restricted to roadways and the pace of movement is usually limited to the maximum allowable speed of the roadway. The motion of mobile nodes in MANETs, on the other hand, is often unconstrained, although it may be restricted by terrain characteristics and physical limitations. Note also that vehicular movements have higher accelerations.

This section presents a vehicular mobility prediction algorithm that probabilistically predicts the near-future movement of the target vehicle based on its current location and estimated speed, assuming that the target has already been localized. The algorithm incorporates: (i) time prediction, which estimates the time elapsed before the target reaches the next intersection; and (ii) direction prediction, which identifies the direction that the target will most likely take after passing the intersection.

6.1 Time Prediction

Predicting the time taken for a vehicle to reach the next intersection requires an estimate of the speed of the vehicle. The estimation procedure is illustrated in Figure 1. Vehicle C_a estimates the speed of C_s by making two RSS measurements at times t_1 and t_2. For reasons of simplicity, it is assumed that the distance between the horizontally-aligned vehicles in a roadway is I, which is easily obtained. However, it is very unlikely that vehicles C_a and C_s will be aligned perfectly. Consequently, when the first RSS measurement (RSS_1) is taken, the vertical distance ahead or behind C_a is calculated to be perfectly adjacent to C_s. The



I
N
C_s
C_a
$d_1(t_1)$
(a) C_a and C_s at time t_1
C_{a+N}
I
C_s
(b) C_a adjusted to C_{a+N} at time t_1
I
C_s
M
N
$d_{2+N}(t_2)$
$d_2(t_2)$
F
E
(c) C_a and C_s at time t_2

Figure 1. Estimating the speed of vehicle C_s.

position of C_a is then adjusted so that it is aligned with C_s using:

$$N = \sqrt{(d_1(t_1))^2 - I^2}$$

as shown in Figure 1(a). Thus, the position C_a becomes C_{a+N} as shown in Figure 1(b).

After the position of C_a is adjusted, the second RSS measurement (RSS_2) is taken at time t_2. However, the distance obtained from RSS_2 must be adjusted by recalculating it as if it was measured by C_{a+N}, which accounts for the extra distance N.

Figure 1(c) illustrates this process. First, the distance M is calculated as:

$$M = \sqrt{(d_2(t_2))^2 - I^2} - N$$

and thus,

$$d_{2+N}(t_2) = \sqrt{M^2 + I^2}.$$

Next, the speed of C_s is computed as:

$$v_s = \frac{F + E}{t_2 - t_1} \tag{1}$$

where $F = \sqrt{(d_{2+N}t_2))^2 - I^2}$ and $E = v_a(t_2 - t_1)$ assuming that the speed v_s of C_a is known (v_s is estimated easily).

After the speed is estimated, the time taken for C_s to reach the next intersection is calculated using the traditional distance equation:

$$t_z = n - d_x + \epsilon/v_s$$

where d_x is the distance the vehicle has covered on a roadway of length n and can be calculated relative to the location of C_a; ϵ is a uniformly distributed random variable introduced to compensate for the acceleration/deceleration error margin; and v_s is the estimated speed of vehicle C_s according to Equation 1.

It is very unlikely that vehicles C_a and C_s have perfect horizontal alignment; thus, we assume that one vehicle is leading or lagging the other. In Figure 1, vehicle C_s is leading C_a and is moving faster than C_a, so it will continue to lead vehicle C_a. Note that the algorithm accommodates other scenarios where C_a is the leading vehicle. This is because the absolute velocity of C_s is measured without considering its relation to the location and speed of C_a.

Since localization is performed before the prediction algorithm is executed, the lane that the target occupies is known. The discussion above assumes that C_a and C_s are not in the same lane, but the probability of them being in the same lane is equally likely. If this is the case and if C_a makes the first RSS measurement (RSS_1) and later makes the second RSS measurement (RSS_2), then the distance C_s has moved during the period between these two measurements is simply the difference between RSS_2 and RSS_1. Moreover, if the vehicles are not in the same lane, it does not matter which vehicle is in which lane. In Figure 1, it happens that C_a is in the left lane and C_s is in the right lane, but the algorithm still works if the lane positions are reversed. However, to simplify the design of the algorithm, we assume that the leading entity will continue to lead during the period between the RSS_1 and RSS_2 measurements. This is a reasonable assumption because the time period is usually short enough to preserve the leading status.

6.2 Direction Prediction

Predicting the direction that a target vehicle will take through the next intersection is more complex and bears a probabilistic distribution. Also, it presupposes that the target vehicle has been accurately localized in order to identify the lane it occupies. We assume that all drivers adhere to the following basic traffic rules as illustrated in Figure 2:

- **Rule 1:** For a vehicle to turn right at an intersection, it must be in the right lane of the roadway leading to the intersection.

- **Rule 2:** For a vehicle to turn left at an intersection, it must be in the left lane of the roadway leading to the intersection.

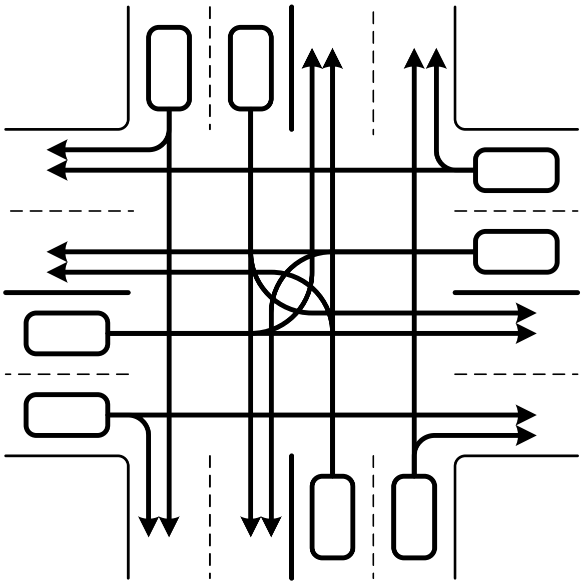


Figure 2. Traffic rules at an intersection.

- **Rule 3:** For a vehicle to go straight ahead at an intersection, it can be in the right lane or in the left lane of the roadway leading to the intersection.

- **Rule 4:** A vehicle must position itself in the appropriate lane based on its intended direction and Rules 1–3 around 300 meters before the next intersection or halfway through the roadway leading to the next intersection, whichever comes later. A lane change beyond this point for the purpose of accelerating the pace (e.g., to overtake a vehicle) is not permitted.

It is reasonable to assume that the target vehicle will obey these traffic rules because the target (who is either a suspect or criminal) will most likely not want to attract attention. Obeying Rules 1–4 forces a vehicle to start the lane changing process when reaching approximately $n/4$ of a roadway of length n or 400 meters away from the intersection, whichever is later. This should allow sufficient time for the vehicle to complete the lane changing process without violating Rule 4. We call the region between $n/4$ and $n/2$ (or similarly, the 100 meters between 400 meters to 300 meters away from the intersection) a "critical area" and apply the following predictions:

$$IF \quad C_i^L \xrightarrow{CriticalArea} C_i^R \quad THEN \quad Pr(C_i \rightarrow R) = 80\%$$
$$IF \quad C_i^R \xrightarrow{CriticalArea} C_i^L \quad THEN \quad Pr(C_i \rightarrow L) = 70\%$$

Based on these predictions, if a particular vehicle C_i occupied the left lane (C_i^L) and shifted to the right lane as it passed through the critical area, the algorithm infers that the shift was imposed by traffic regulations. This means that the vehicle is very likely going to turn right (R) at the next intersection because: (i) the probability of turning left (L) from a right lane (where the vehicle has shifted) is 0; and (ii) if the vehicle intended to go straight ahead (A), it would most likely have stayed in its original lane without taking the overhead of a lane change as it is permitted to go straight ahead from either lane.

A similar argument applies to a vehicle that shifted from the right lane to the left lane as it passed through the critical area. The difference is that it is possible that the vehicle shifted from the right lane to the left lane to increase its pace of the movement while actually intending to go straight ahead. This decreases the probability of turning left for this particular course of action.

On the other hand, if the vehicle did not change its lane as it went through the critical area, it has a probability of 50% of taking either of the two permitted directions depending on its current lane. Also, it has a 0% probability of taking the banned direction (i.e., left for the right lane and right for the left lane).

An interesting extension to this direction prediction technique (which is not currently implemented in the algorithm) is to integrate points of interest [5] corresponding to frequently-visited locations such as grocery stores, banks, restaurants and offices. Such locations can significantly influence the probability distribution of vehicular mobility. For example, we know that the probability of a vehicle in the right lane turning right is (currently) the same as the probability of the vehicle going straight ahead if it had not shifted to the right lane as it passed through the critical area. However, that fact that a point of interest is located on the right could increase the probability of the vehicle turning right instead of going straight ahead. Note that the specific time of day can significantly influence the effect of a point of interest on the prediction probabilities (e.g., an office building will most likely be a popular point of interest only during the daytime).

7. Tracking

In forensic and law enforcement applications, vehicle tracking is usually required to be passive so that the driver of the target vehicle is not aware of the tracking process. This passivity requirement potentially eliminates more active tracking systems such as those involving vehicle telematics.

In our tracking scenario, a group of trackers $C_{t(i)}$ $(i = 1, 2, ..., n)$ consisting of n privately-connected vehicles (e.g., police patrols) recruit a main tracking agent C_a and a backup tracking agent C_b from the public to track a target vehicle C_s. We assume that the recruiting $C_{t(i)}$ is initially located in range of C_s, which enables it to recruit suitable C_a and C_b possibly by visual estimation. During this initial stage, the $C_{t(i)}$ also recruits an additional localization agent C_l which, along with the tracking agents, localize the target as discussed in Section 5. The recruiting $C_{t(i)}$ supplies an address list of the other valid $C_{t(i)}$ to all the recruited agents.

The localizing agent is retired after the target is localized, leaving C_a to be responsible for the rest of the tracking. C_a is backed up by C_b, whose responsibility is to take over the tracking process if C_a suddenly fails. C_a is also responsible for sending location updates of C_s to a $C_{t(i)}$ whenever localization is triggered. C_a must ensure that its backup agent C_b is within range of C_a and the target at all times by regularly monitoring its RSS and probing it for the target's RSS; otherwise it has to recruit another C_b.

When tracking is initiated, the $C_{t(i)}$ creates a tracking table $TrackTB$, which stores records of the target's movements as received from C_a. This table is synchronized with all other $C_{t(i)}$ as soon as an update is received by a $C_{t(i)}$. When a localization update is available, C_a searches its range for a $C_{t(i)}$ to update it; if multiple $C_{t(i)}$ are found, C_a randomly chooses one of them; this potentially minimizes a traffic analysis attack by a third observer who observes the traffic between the agents and the trackers. If no $C_{t(i)}$ is available, C_a creates a temporary tracking table $TempTB$ and accumulates information while regularly probing for a valid $C_{t(i)}$. When a $C_{t(i)}$ is found, C_a transfers its $TempTB$ to the C_{ti} which, in turn, merges it with its copy of $TrackTB$ and synchronizes it with the other $C_{t(i)}$.

After running the prediction algorithm, C_a estimates how long it will be able to track the target (called the "alive period") and schedules the next localization for near the expiration of this period. During the alive period, C_a keeps observing the RSS measurements from C_s. If a specific lower threshold of RSS is reached or the predicted alive time is near expi-

Algorithm 1 Vehicular Tracking Algorithm

```
1:  $C_{t(i)} \Leftarrow$ Trackers {identify $n$ trackers}
2:  $C_s \Leftarrow$ Target {identify the target}
3:  $Recruit(C_a, C_b)$ {recruit tracking agents}
4:  $Recruit(C_l)$ {recruit localization agent}
5:  repeat
6:    $Localize(C_s)$ {$C_a$, $C_b$, $C_l$ Localize $C_s$}
7:    $Retire(C_l)$
8:    $Alive \leftarrow Predict(C_s)$ {find alive period based on the prediction
      algorithm}
9:    while $currentTime \leq Alive - \omega$ do
10:     if $C_a.RSS(C_s) \leq$ threshold then
11:       break {if RSS from $C_s \leq$ threshold, break}
12:     end if
13:   end while
14:   $C_a$.Probe {search for agents}
15:   if $C_a.RSS(C_s) \leq$ threshold then
16:     if $C_b.RSS(C_s) \leq$ threshold then
17:       $Recruit(C_a, C_b,)$ {recruit new $C_a$ and $C_b$}
18:     end if
19:     $Recruit(C_a)$ {recruit new $C_a$}
20:   else
21:     $Recruit(C_l)$ {recruit $C_l$}
22:   end if
23: until Tracking Expires
```

ration, C_a initiates a "probe process," which involves sending requests to the neighboring vehicles, supplying the C_s address and asking for their RSS measurements if they can receive transmissions from C_s. Note that the vehicle with the strongest RSS is recruited as the localization agent C_l. Furthermore, a $C_{t(i)}$ may at any time request C_a to localize C_s, in which case C_a executes the probe process to recruit C_l and works with C_b to localize C_s.

If C_a had to maintain a $TempTB$, but attempted to recruit another C_a (due to the expiry of the alive period or weak RSS observations from C_s) and there is still no $C_{t(i)}$ in range, the old C_a recruits a new C_a and hands off the tracking process to it along with the $TempTB$. The $TempTB$ is maintained by the new C_a until a valid $C_{t(i)}$ is found upon which time the old C_a retires itself. Details of the tracking process are presented in Algorithm 1.

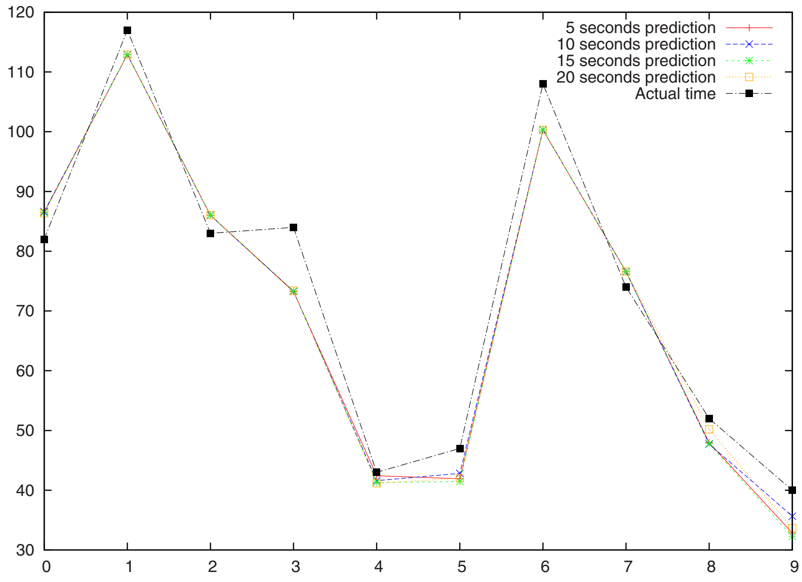


Figure 3. Simulation results (10 node scenario).

8. Simulation Results

A realistic vehicular environment was created using the VanetMobiSim simulator [7]; vehicular mobility traces were generated based on IDM-LC model. The ns-2 simulator was used to run the tracking scenarios.

The prediction algorithm was evaluated by executing it over a number of different scenarios and estimating the time taken for vehicles to reach the next intersection. These times were compared with the actual times taken to reach the intersection as reported by the network simulator. The simulation scenarios adopted a set of interconnected roadways with different lengths and speed limits. Additionally, mobility traces with different node densities were generated in order to analyze the effect of node density on the accuracy of the prediction algorithm.

Figures 3 and 4 present the results of the prediction algorithm for node densities of 10 and 100 nodes with the simulations running for 1,000 seconds. Clearly, there is a variation in performance. The main factors influencing the precision of the prediction algorithm are:

- **Node Density:** As seen in Figures 3 and 4, the accuracy of the algorithm is moderately affected by increased node density. This is expected in a real-world environment – the more vehicles present

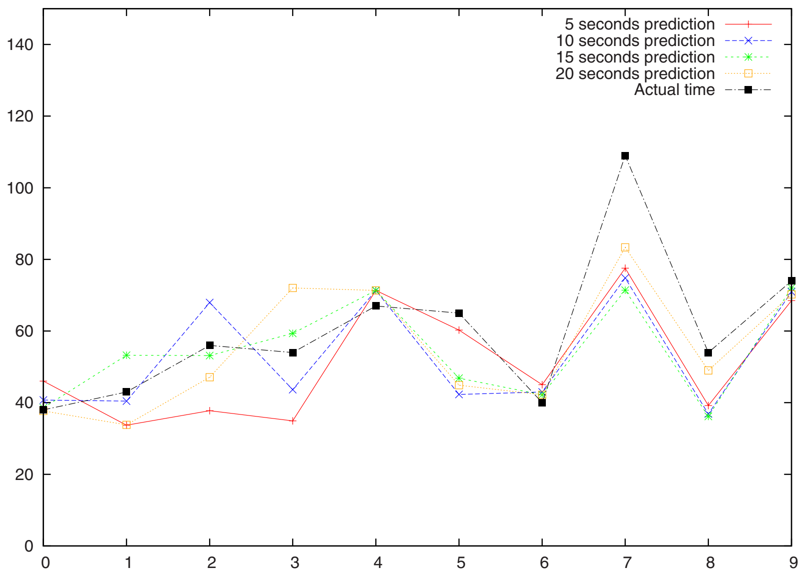


Figure 4. Simulation results (100 node scenario).

on a roadway, the harder it is to accurately predict their movement due to their irregular acceleration/deceleration behaviors.

- **Roadway Length:** The length of the roadway affects the prediction algorithm because longer roadways allow for more fluctuations in acceleration, which can be difficult to model. Highways have very different characteristics than city roadways in terms of traffic flow.

- **Speed Limit:** Roadways with higher speed limits also allow for increased acceleration fluctuations, which affect the accuracy of the algorithm.

- **Duration of the Prediction:** Other than the macroscopic factors above, the duration that the prediction process is applied to the target is an important factor. The prediction algorithm measures the time taken by the target to pass a specific distance and then adds noise to the measurement to compensate for future acceleration/deceleration before calculating the predicted time. However, the accuracy of the algorithm is improved when the target is observed for a longer interval. This is illustrated in Figures 3 and 4, where a longer prediction interval of 20 seconds (i.e., the target's movement is observed for 20 seconds) yields a better prediction than a shorter interval. Another way to improve prediction accuracy is to take multiple observations at different times and compute the average.

- **Location of the Prediction:** Despite the fact that the prediction process can take place at any point on the roadway, the point where the prediction process takes place can be important, especially for highways that typically have higher speed limits than city roadways. For example, it is not advisable to execute the prediction process at the beginning of a highway. Since vehicles just start accelerating at this point, their accelerations are not representative of the rest of the journey.

Based on the above factors, it is clear that the tracking environment has an unavoidable influence on the tracking process. Carefully modeling the macroscopic features of the environment is, therefore, critical to improving tracking accuracy. Information about the general structure and layout of roadways can be obtained from online databases such as TIGER [11]. Other important road characteristics (e.g., density) can be modeled and estimated based on empirical observations.

9. Conclusions

Vehicular tracking has important applications in crime prevention and criminal investigations. The algorithms presented in this paper implement the passive localization, tracking and prediction of the movement of a target vehicle. While there has been limited research related to passive vehicular tracking, we argue that the passivity requirement is very important for forensic and law enforcement purposes. Note, however, that the agent-based approach may face barriers in some jurisdictions where randomly recruiting tracking agents may be restricted by law.

Finally, we note that the RF measurements used in the algorithms include an unavoidable error margin. However, estimating the measurement parameters can provide an acceptable error margin. This is because in vehicular tracking it is only important to detect the presence of the target and partially track it to estimate its most likely trace in the corresponding roadway. Inferences can then be made if the target halted en route (e.g., to commit a crime) by observing the estimated time delay along each roadway.

References

[1] R. Baumann, S. Heimlicher and M. May, Towards realistic mobility models for vehicular ad hoc networks, *Proceedings of the Twenty-Sixth IEEE Conference on Computer Communications*, 2007.

[2] A. Benslimane, Localization in vehicular ad hoc networks, *Proceedings of the Systems Communications Conference*, pp. 19–25, 2005.

[3] A. Boukerche, H. Oliveira, E. Nakamura and A. Loureiro, Vehicular ad hoc networks: A new challenge for localization-based systems, *Computer Communications*, vol. 31(12), pp. 2838–2849, 2008.

[4] S. Brakatsoulas, D. Pfoser, R. Salas and C. Wenk, On map-matching vehicle tracking data, *Proceedings of the Thirty-First International Conference on Very Large Data Bases*, pp. 853–864, 2005.

[5] D. Engelhart, A. Sivasubramaniam, C. Barrett, M. Marathe, J. Smith and M. Morin, A spatial analysis of mobility models: Application to wireless ad hoc network simulation, *Proceedings of the Thirty-Seventh Annual Symposium on Simulation*, p. 34, 2004.

[6] H. Friis, A note on a simple transmission formula, *Proceedings of the IRE*, vol. 34(5), pp. 254–256, 2006.

[7] J. Harri, F. Filali, C. Binnet and M. Fiore, VanetMobiSim: Generating realistic mobility patterns for VANETs, *Proceedings of the Third International Workshop on Vehicular Ad Hoc Networks*, pp. 96–97, 2006.

[8] J. Luo and J. Hubaux, A Survey of Inter-Vehicle Communication, Technical Report IC/2004/24, School of Computer and Communication Sciences, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 2004.

[9] J. Mansell and W. Riley, Vehicle Tracking and Security System, United States Patent 5,233,844, 1993.

[10] M. Treiber, A. Hennecke and D. Helbing, Congested traffic states in empirical observations and microscopic simulations, *Physical Review E*, vol. 62(2), pp. 1805–1824, 2000.

[11] U.S. Census Bureau, Topologically Integrated Geographic Encoding and Referencing, Washington, DC (www.census.gov/geo/www /tiger).

Chapter 8

A FORENSIC READINESS MODEL FOR WIRELESS NETWORKS

Sipho Ngobeni, Hein Venter and Ivan Burke

Abstract Over the past decade, wireless mobile communications technology based on IEEE 802.11 wireless local area networks (WLANs) has been adopted worldwide on a massive scale. However, as the number of wireless users has soared, so has the possibility of cyber crime, where criminals deliberately and actively break into WLANs with the intent to cause harm or access sensitive information. WLAN digital forensics is seen not only as a response to cyber crime in wireless environments, but also as a means to stem the increase of cyber crime in WLANs. The challenge in WLAN digital forensics is to intercept and preserve all the communications generated by the mobile devices and conduct a proper digital forensic investigation. This paper attempts to address this issue by proposing a wireless forensic readiness model designed to help monitor, log and preserve wireless network traffic for digital forensic investigations. A prototype implementation of the wireless forensic readiness model is presented as a proof of concept.

Keywords: Wireless local area networks, digital forensic readiness

1. Introduction

Wireless technologies have become very popular around the world. Wireless local area networks (WLANs) or "hotspots" blanket public places such as convention centers, airports, schools, hospitals, railway stations, coffee shops and other locations to provide seamless public access to the Internet [15]. These hotspots provide several advantages over hard-wired networks, including user mobility and flexible Internet access. However, due to their open nature, WLANs have become a major target for cyber criminals.

WLAN digital forensics involves the application of methodologies and tools to intercept and analyze wireless network events for presentation as digital evidence in a court of law [9]. As such, WLAN digital forensics is complementary to intrusion prevention – when intrusion prevention fails, WLAN digital forensics is useful for obtaining information about the intrusion. However, the primary challenge in WLAN digital forensics is to acquire all the digital evidence related to a crime [6]. This challenge arises from the fact that the devices participating in a WLAN environment are mobile. Furthermore, since the devices are not always connected to the network, it is difficult to attribute criminal activity to a particular device.

This paper proposes a wireless forensic readiness model for monitoring, logging and preserving wireless network traffic for digital forensic investigations. The wireless forensic readiness model builds on the work of Rowlingson [7] related to traditional forensic investigations. A prototype implementation of the readiness model is presented as a proof of concept.

2. Wireless Local Area Networks

WLANs represent a ubiquitous technology that provides seamless high-speed Internet connectivity at public locations. Unlike traditional LANs, WLANs connect computers to the network without physical (wired) connections. WLANs offer tremendous user mobility, enabling users to access files, network resources and the Internet [8].

2.1 Criminal Misuse of WLANs

The lack of a physical connection between a WLAN and its participating mobile devices causes crimes to remain discreet, especially since the mobile devices are potentially far removed. This fact needs to be considered when digital evidence is identified and collected in an investigation involving wireless traffic. Potential criminal misuse of WLANs include [12, 14]:

- **WLAN Detection and Connection:** This type of misuse involves an intruder using the wireless medium as a tool to commit other criminal activities (e.g., unauthorized use of the WLAN or use of the WLAN as a launch pad for other criminal activities).

- **Concealment of Digital Evidence:** This type of misuse involves hidden wireless devices or hidden wireless networks (e.g., fake access points).

- **WLAN as an Attack Vector:** This type of misuse involves attacks against the networked (mobile) devices originating from the wireless network, and attacks against the WLAN medium itself.

2.2 Sources of Digital Evidence

WLANs typically incorporate 802.11-based wireless devices. The locations where digital evidence is stored on devices and the extraction of evidence are dependent on the specific wireless device. However, the fundamental problem with 802.11-based wireless devices is their lack of a physical footprint, which is the most crucial issue in the identification of these devices [14].

It is imperative to locate all the relevant wireless devices in a digital forensic investigation. Various open source and commercial tools (e.g., Wireshark, Kismet and AirCapture) may be used by a digital forensic investigator to identify wireless networks within range, the devices connected to the wireless networks and, possibly, the locations of the wireless devices [13]. The principal drawback of these tools is the high packet drop rate [1]. The large volume of network traffic makes it difficult for the tools to accept and store all the packets; some packets may be dropped, resulting in the loss of evidence.

2.3 WLAN Digital Forensics

Digital forensics deals with the investigation of computers and other digital devices believed to be involved in criminal activities [5]. WLAN digital forensics involves the application of methodologies and tools to capture and analyze wireless network traffic that can be presented as evidence in a court of law [9]. A WLAN digital forensic methodology is a digital forensic process; the tools are software systems that intercept and analyze network traffic. A digital forensic process is a procedure that is followed to investigate a particular criminal activity involving digital evidence [2]. Every digital forensic investigation must go through the following phases of the digital forensic process:

- Define the scope and goals of the investigation.
- Determine the work and materials.
- Acquire the images of the devices to be examined.
- Perform the digital forensic analysis.
- Prepare the report.

Currently, EnCase and FTK are the most popular tools used in digital forensic investigations. The phases of the digital forensic process for EnCase are preview, imaging or acquisition, verification, recovery and analysis, restoration and archiving; the phases for FTK are detection,

Table 1. Digital forensic phases for EnCase, FTK and WFRM.

EnCase	**FTK**	**WFRM**
1. Preview	1. Detection	1. Monitoring
2. Imaging	2. Identification	2. Logging
3. Verification	3. Analysis	3. Preservation
4. Recovery and Analysis	4. Preservation	4. Analysis
5. Restoration	5. Reporting	5. Reporting
6. Archiving		

identification, analysis, preservation and reporting. Table 1 lists the phases for EnCase and FTK along with those for the wireless forensic readiness model (WFRM), which is described in the next section.

According to Table 1, only the analysis phase is common to EnCase, FTK and WFRM. The preservation and analysis phases are common to FTK and WFRM. However, it is worth noting that the digital forensic processes for FTK and EnCase are essentially the same as far as the general digital forensic process is concerned. This suggests that the phases correlate although they are named differently. The inconsistent naming of phases is due to the fact that the digital forensic processes for forensic tools are not standardized. In this paper, we adopt the general digital forensic process described by Casey [2].

Researchers have studied various issues related to wireless network forensics. Yim, *et al.* [16] proposed a WLAN forensic profiling system for collecting digital evidence after denial-of-service attacks on WLANs. Turnbull and Slay [14] consider the potential sources of digital evidence in 802.11-based wireless networking environments. Then [11] discusses methods for examining wireless access points to determine if the devices of interest are connected or were connected to a wireless network. While these efforts and others are useful, a digital forensic readiness approach for WLANs has yet to be articulated.

2.4 Digital Forensic Readiness

The purpose of digital forensic readiness is to reduce the effort involved in performing an investigation while maintaining the level of credibility of the digital evidence being collected [4]. The decrease in effort includes reductions in the time and the cost of incident response. An organization that is "forensically ready" can respond to an attack rapidly and efficiently. In general, reducing the time involved in incident response reduces the cost of the investigation.

Tan [10] discusses an incident in which an intruder took approximately two hours to launch an attack, but digital forensic experts required 40 billable hours to respond to the incident. The response took such a long time because the organization was not forensically prepared for the incident. Organizations deploying WLANs that are at a high risk of cyber attack should be ready to collect digital evidence before an incident occurs. The model presented in the next section addresses the concept of digital forensic readiness in WLANs.

3. Wireless Forensic Readiness Model

The most salient characteristic of the wireless forensic readiness model (WFRM) is that it monitors wireless network traffic at access points. The monitored traffic is stored in a log file and the integrity of the stored information is preserved. Thus, the information needed by digital forensic investigators is readily available should the need arise. The availability of the information reduces the cost of conducting the digital forensic investigation because a major portion of the digital forensic process (monitoring, logging and preservation) has already been conducted based on the WFRM.

Figure 1 shows the five phases of the digital forensic process corresponding to the WFRM. As listed in Table 1, the phases are monitoring, logging, preservation, analysis and reporting.

Phase 1 (monitoring) shows several mobile devices (MDs) connected to a WLAN through different access points (APs). The mobile devices use the access points to connect to the Internet. In addition to providing Internet connectivity, the access points are modified (for the purposes of this model) to monitor all the traffic generated by the mobile devices. For security reasons, the monitoring component uses a firewall to filter inbound and outbound wireless traffic. Filtering is the process of controlling access to the WLAN by screening packets based on the content of their headers [15].

Phase 2 (logging) records all the traffic monitored by the access points. Each access point has its own capture unit (CU) that logs the traffic passing through it. The log file is divided into separate storage areas, each consisting of (for example) 1 MB of data. When the buffer of a capture unit is full, a fixed-size block of data is moved to permanent storage. For example, B1 in Phase 2 represents a block of data consisting of 4 MB.

In Phase 3 (preservation), the capture unit sends the accumulated blocks of data to the evidence store (ES). The capture unit computes a hash value for each block of data, which is saved in the hash store

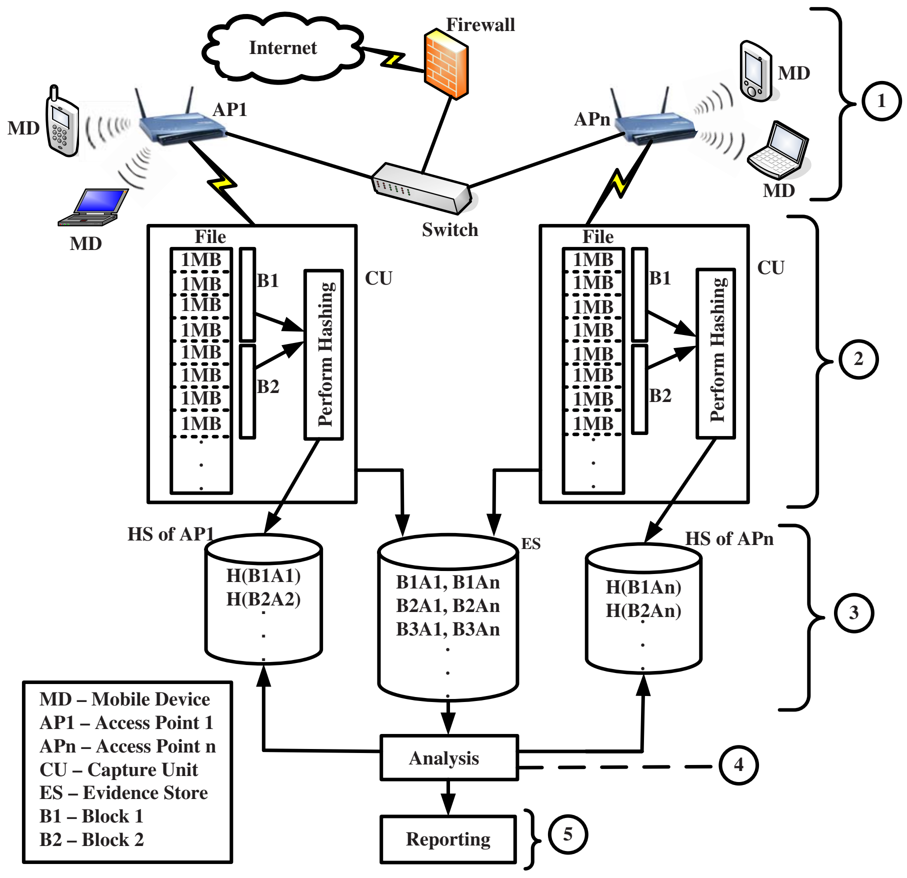


Figure 1. Wireless forensic readiness model (WFRM).

(HS) for integrity checking purposes. Phase 4 involves the analysis of the stored data and Phase 5 involves the creation of a report.

4. WFRM Simulation

The WFRM prototype was simulated using AnyLogic Professional (version 6.0) [3], a Java-based, multi-paradigm, hybrid simulation tool capable of modeling systems as a combination of discrete events, system dynamics and agents. The simulation is designed to validate the use of the WFRM for implementing digital forensic readiness in WLAN environments.

Figure 2 shows a graphical representation of the WFRM before the simulation starts. The *MobileDevice* component generates random simu-

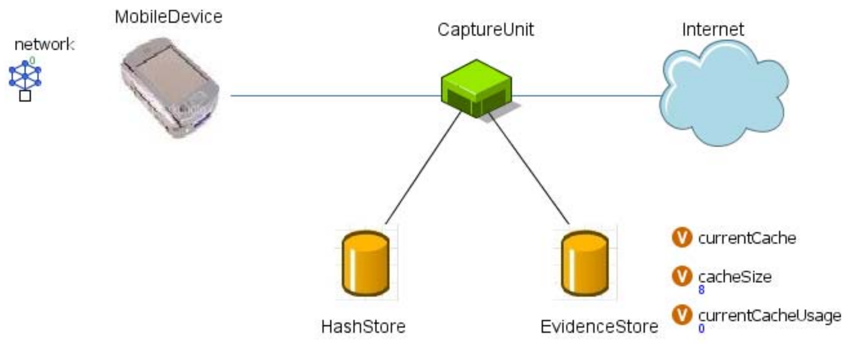


Figure 2. WFRM before the simulation.

lated messages containing source and destination IP addresses, message transmission date and time, and message content.

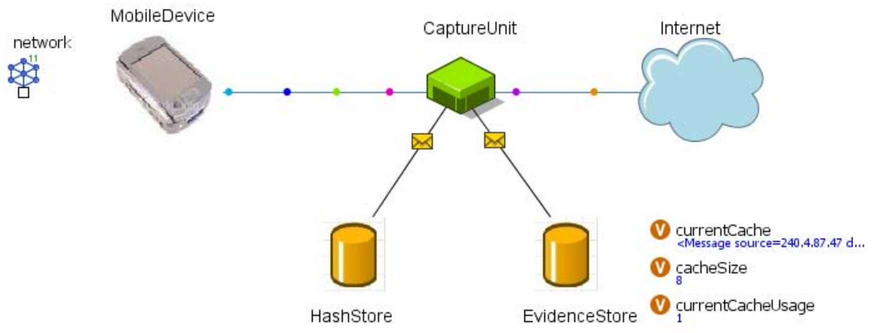
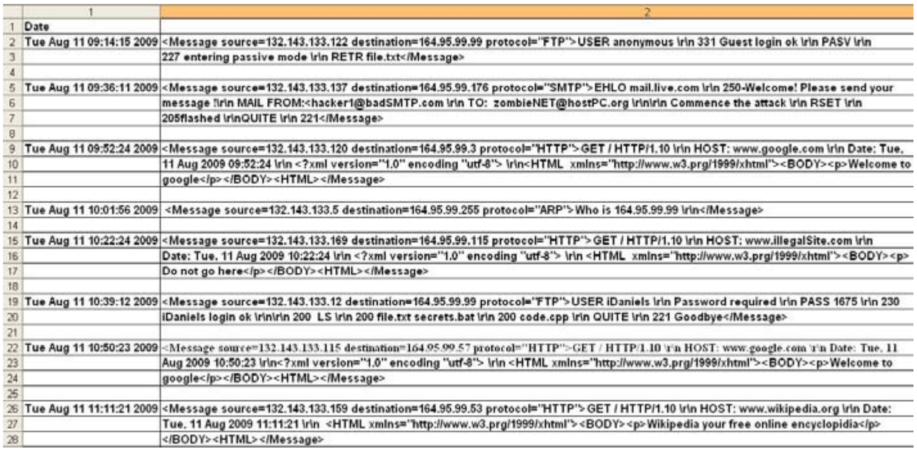


Figure 3. WFRM during the simulation.

Figure 3 shows the simulated messages flowing through the network during the simulation. The simulated messages correspond to the packets that pass through the network from various devices in the WLAN.

CaptureUnit contains the variables *currentCache*, *cacheSize* and *currentCacheUsage*. The *currentCache* variable represents the log file in our model, which works like a buffer; *currentCache* can store up to eight packets (based on the *cacheSize* in Figure 3). The eight captured packets are put together to form a single message; this represents a created block of data in our model. *CaptureUnit* computes the hash value of the formed message and stores the value in the *HashStore*; also, it passes the



	1	2
1	Date	
2	Tue Aug 11 09:14:15 2009	<Message source=132.143.133.122 destination=164.95.99.99 protocol="FTP">USER anonymous \r\n 331 Guest login ok \r\n PASV \r\n
3		227 entering passive mode \r\n RETR file.txt</Message>
4		
5	Tue Aug 11 09:36:11 2009	<Message source=132.143.133.137 destination=164.95.99.176 protocol="SMTP">EHLO mail.live.com \r\n 250-Welcome! Please send your
6		message !\r\n MAIL FROM:<hacker1@badSMTP.com \r\n TO: zombieNET@hostPC.org \r\n\r\n Commence the attack \r\n RSET \r\n
7		205flashed \r\nQUITE \r\n 221</Message>
8		
9	Tue Aug 11 09:52:24 2009	<Message source=132.143.133.120 destination=164.95.99.3 protocol="HTTP">GET / HTTP/1.10 \r\n HOST: www.google.com \r\n Date: Tue.
10		11 Aug 2009 09:52:24 \r\n <?xml version="1.0" encoding "utf-8"> \r\n<HTML xmlns="http://www.w3.prg/1999/xhtml"><BODY><p>Welcome to
11		google</p></BODY><HTML></Message>
12		
13	Tue Aug 11 10:01:56 2009	<Message source=132.143.133.5 destination=164.95.99.255 protocol="ARP">Who is 164.95.99.99 \r\n</Message>
14		
15	Tue Aug 11 10:22:24 2009	<Message source=132.143.133.169 destination=164.95.99.115 protocol="HTTP">GET / HTTP/1.10 \r\n HOST: www.illegalSite.com \r\n
16		Date: Tue. 11 Aug 2009 10:22:24 \r\n <?xml version="1.0" encoding "utf-8"> \r\n <HTML xmlns="http://www.w3.prg/1999/xhtml"><BODY><p>
17		Do not go here</p></BODY><HTML></Message>
18		
19	Tue Aug 11 10:39:12 2009	<Message source=132.143.133.12 destination=164.95.99.99 protocol="FTP">USER iDaniels \r\n Password required \r\n PASS 1675 \r\n 230
20		iDaniels login ok \r\n\r\n 200 LS \r\n 200 file.txt secrets.bat \r\n 200 code.cpp \r\n QUITE \r\n 221 Goodbye</Message>
21		
22	Tue Aug 11 10:50:23 2009	<Message source=132.143.133.115 destination=164.95.99.57 protocol="HTTP">GET / HTTP/1.10 \r\n HOST: www.google.com \r\n Date: Tue. 11
23		Aug 2009 10:50:23 \r\n<?xml version="1.0" encoding "utf-8"> \r\n <HTML xmlns="http://www.w3.prg/1999/xhtml"><BODY><p>Welcome to
24		google</p></BODY><HTML></Message>
25		
26	Tue Aug 11 11:11:21 2009	<Message source=132.143.133.159 destination=164.95.99.53 protocol="HTTP">GET / HTTP/1.10 \r\n HOST: www.wikipedia.org \r\n Date:
27		Tue. 11 Aug 2009 11:11:21 \r\n <HTML xmlns="http://www.w3.prg/1999/xhtml"><BODY><p>Wikipedia your free online encyclopidia</p>
28		</BODY><HTML></Message>

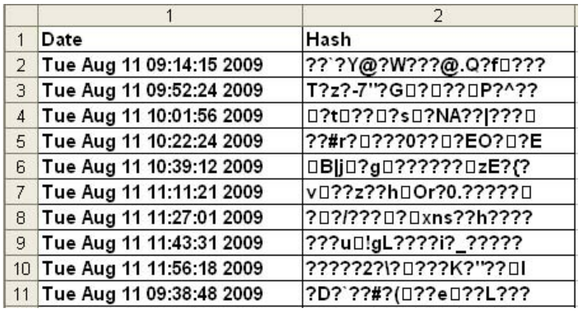
Figure 4. Evidence store.

formed message to the *EvidenceStore* for storage. The variable *currentCacheUsage* keeps track of the number of times that *currentCache* was filled with the eight packets that are combined to form a single message.

Figure 4 presents sample data in *EvidenceStore*. The message in Row 2 shows that an anonymous user with IP address `132.143.133.122` is attempting to log into a remote host via FTP. The fact that this machine is anonymous could be of interest to a digital forensic investigator. The message in Row 5 contains data such as "Please send your message," "hacker1@badSMTP.com" and "zombieNET@hostPC.org." This data seems suspicious and constitutes potential digital evidence.

The message in Row 9 shows that a machine is using HTTP to access Google; this does not appear to indicate any malicious activity. The message in Row 13 shows that the machine with IP address X is asking the machine with IP address Y if it knows the machine with IP address Z; this does not appear to be malicious. However, if Machine Y responds to Machine X that it is not aware of Machine Z, then it is possible that Machine Z is not part of the network and could be an intruder who intends to sniff network traffic between Machines X and Y. The message in Row 15 shows that a machine with IP address `132.143.133.169` is accessing a suspicious website named "www.illegalSite.com;" an error message "Do not go here" pops up when this website is accessed. The message in Row 19 shows that a machine is providing its login details to a website and downloading a suspicious file named `secrets.bat`.

Figure 5 presents the *HashStore* corresponding to the captured simulated messages. Every time a message is captured, a copy of the original



	1	2
1	Date	Hash
2	Tue Aug 11 09:14:15 2009	??`?Y@?W???@.Q?f□???
3	Tue Aug 11 09:52:24 2009	T?z?-7"?G□?□??□P?^??
4	Tue Aug 11 10:01:56 2009	□?t□??□?s□?NA??|???□
5	Tue Aug 11 10:22:24 2009	??#r?□???0??□?EO?□?E
6	Tue Aug 11 10:39:12 2009	□B|j□?g□??????□zE?{?
7	Tue Aug 11 11:11:21 2009	v□??z??h□Or?0.?????□
8	Tue Aug 11 11:27:01 2009	?□?/???□?□xns??h????
9	Tue Aug 11 11:43:31 2009	???u□!gL????i?_?????
10	Tue Aug 11 11:56:18 2009	?????2?\?□???K?"??□I
11	Tue Aug 11 09:38:48 2009	?D?`??#?(□??e□??L???

Figure 5. Hash store.

captured message is hashed and transferred to the *HashStore.* The main reason for hashing the captured information and keeping a separate copy of the original information is to verify the integrity of the captured information and to determine whether or not it was tampered with. The integrity of any message can be verified by extracting and computing the hash value (y) of the message stored in the *EvidenceStore.* The hash value (y) for the particular message block is then retrieved from the *HashStore.* If the hash values x and y match, the content of the original captured message was not tampered with and the integrity of the captured message in the *EvidenceStore* is verified. The integrity checking mechanism was built into the prototype because the integrity of evidence is a crucial requirement in any digital forensic investigation [2].

5. Discussion

In the simulation described in the preceding section, the simulated packets were logged (by *CaptureUnit*) and preserved (by *EvidenceStore* and *HashStore*). However, note that traffic monitoring was not implemented in the prototype because it is performed by the access point mainly for security reasons. After the traffic generated by the mobile devices that have connected to the WLAN has been captured and preserved, the data is ready for analysis in a digital forensic investigation. Because this data is forensically ready and forensically sound, the time and cost involved in conducting the digital forensic investigation are reduced considerably. In fact, the data needed for the investigation is readily available and the bulk of the digital forensic process (i.e., monitoring, logging and preservation) has been completed.

One disadvantage of the WFRM simulation is that the traffic is preserved in the *EvidenceStore* and *HashStore*, which potentially requires a large amount of storage space. This is not a serious problem because storage is becoming ever cheaper. Nevertheless, we are working on compression techniques that will facilitate the preservation of the entire stream of wireless network traffic. Since this might not be an optimal long-term solution to the problem, further research is needed to address the storage issue.

Finally, we note that the digital forensic processes for EnCase, FTK and WFRM (Table 1) are essentially equivalent. However, since our emphasis in this paper is the design of a readiness model, the practical implementation of the digital forensic process employed for WFRM is different from the conventional digital forensic process models for EnCase and FTK.

6. Conclusions

The wireless forensic readiness model helps address the twin challenges of intercepting and preserving all the communications generated by mobile devices in WLANs. In general, WLANs are not forensically prepared to gather digital evidence for use in ensuing investigations. The forensic readiness model focuses on the monitoring, logging and preservation of wireless network traffic. This covers the bulk of the general digital forensic investigation process, reducing both the time and the cost of forensic investigations.

Our future research will focus on several issues. One issue, as mentioned above, is the efficient storage of data in the hash store and evidence store. Another key issue is the analysis of potentially large amounts of data gathered as a result of the application of the wireless forensic readiness model. Other issues involve evidence management and the consideration of infrastructure requirements, admissibility requirements and retention requirements.

References

[1] J. Broadway, B. Turnbull and J. Slay, Improving the analysis of lawfully intercepted network packet data captured for forensic analysis, *Proceedings of the Third International Conference on Availability, Reliability and Security*, pp. 1361–1368, 2008.

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

[3] Coensys, AnyLogic 6: Multi-Paradigm Simulation Software, Cherry Hill, New Jersey (www.coensys.com/anylogic.htm).

[4] B. Endicott-Popovsky, D. Frincke and C. Taylor, A theoretical framework for organizational network forensic readiness, *Journal of Computers*, vol. 2(3), pp. 1–11, 2007.

[5] G. Francia and K. Clinton, Computer forensics laboratory and tools, *Journal of Computing Sciences in Colleges*, vol. 20(6), pp. 143–150, 2005.

[6] R. Newman, *Computer Forensics: Evidence Collection and Management*, Auerbach Publications, Boca Raton, Florida, 2007.

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

[8] K. Scarfone, D. Dicoi, M. Sexton and C. Tibbs, Guide to Securing Legacy IEEE 802.11 Wireless Networks, NIST Special Publication 800-48, Revision 1, National Institute of Standards and Technology, Gaithersburg, Maryland, 2008.

[9] R. Siles, Wireless forensics: Tapping the air – Part one, Symantec Corporation, Mountain View, California (www.securityfocus.com /infocus/1884), 2007.

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

[11] C. Then, Examining wireless access points and associated devices, Forensic Focus (www.forensicfocus.com/downloads/examining-wire less-access-points.pdf), 2006.

[12] B. Turnbull and J. Slay, The 802.11 technology gap – Case studies in crime, *Proceedings of the IEEE Region 10 Conference*, 2005.

[13] B. Turnbull and J. Slay, Wireless forensic analysis tools for use in the electronic evidence collection process, *Proceedings of the Fortieth Annual Hawaii International Conference on Systems Sciences*, 2007.

[14] B. Turnbull and J. Slay, Wi-Fi network signals as a source of digital evidence: Wireless network forensics, *Proceedings of the Third International Conference on Availability, Reliability and Security*, pp. 1355–1360, 2008.

[15] E. Velasco, W. Chen, P. Ji and R. Hsieh, Wireless forensics: A new radio frequency based location system, *Proceedings of the Pacific-Asia Workshop on Cybercrime and Computer Forensics*, pp. 272–277, 2008.

[16] D. Yim, J. Lim, S. Yun, S. Lim, O. Yi and J. Lim, The evidence collection of DoS attack in WLAN by using WLAN forensic profiling system, *Proceedings of the International Conference on Information Science and Security*, pp. 197–204, 2008.

III

INTERNET CRIME INVESTIGATIONS

Chapter 9

EVALUATION OF EVIDENCE IN INTERNET AUCTION FRAUD INVESTIGATIONS

Michael Kwan, Richard Overill, Kam-Pui Chow, Jantje Silomon, Hayson Tse, Frank Law and Pierre Lai

Abstract Internet auction fraud has become prevalent. Methodologies for detecting fraudulent transactions use historical information about Internet auction participants to decide whether or not a user is a potential fraudster. The information includes reputation scores, values of items, time frames of various activities and transaction records. This paper presents a distinctive set of fraudster characteristics based on an analysis of 278 allegations about the sale of counterfeit goods at Internet auction sites. Also, it applies a Bayesian approach to analyze the relevance of evidence in Internet auction fraud cases.

Keywords: Internet auction fraud, Bayesian network, relevance of evidence

1. Introduction

According to the Data Center of the China Internet [5], Chinese users spent 2.56 trillion Renminbi ($698 billion) on the Internet during the first half of 2008, a 58.2% increase over the same period in 2007. Of the total amount, 35% was spent on purchases made via the Internet; the remaining 65% was spent on on-line games and network communities. China already has more Internet users than any other country in the world, and the number of users is expected to nearly double from 253 million in 2008 to 480 million in 2010 [2]. By 2010, the volume of online transactions in China will exceed those in Japan and South Korea [2].

Internet auctions offer buyers unparalleled selections of products and the opportunity to make great deals. They also provide sellers with a means to reach millions of potential buyers. Meanwhile, criminals are attracted by the low entry costs and tremendous profits of Internet auctions. Unscrupulous sellers take advantage of buyers by misrepresenting the quality or condition of their goods. Some have no intention of delivering the goods that are offered for sale. As a result, Internet auction fraud is the most common type of fraud reported in the electronic commerce domain [16].

This paper examines the characteristics of Internet auction fraud in Hong Kong related to the sale of counterfeit goods (i.e., goods bearing false trade descriptions or forged trademarks). In addition, it uses Bayesian network models representing the prosecution and defense viewpoints in conjunction with the likelihood ratio as a criterion to determine the relevance of digital evidence in Internet auction fraud cases.

2. Background and Related Work

This section reviews the nature of Internet auction fraud. Also, it surveys approaches for detecting fraud in online auctions.

2.1 Internet Auctions

Internet auctions are successful for many reasons. Potential buyers have sufficient time to search for items of interest and they can bid for items 24 hours a day, seven days a week. The Internet does not impose geographical constraints on buyers and sellers and they are not required to be physically present at an auction. The large numbers of buyers and sellers tend to reduce selling costs as well as sales prices. Many users describe their online auction experience as comparable to gambling. Offering the highest bid provides the same thrill as winning a game.

Ochaeta [16] lists six basic features of Internet auctions:

- **Initial Buyer and Seller Registration:** This step helps authenticate the trading parties. It involves the exchange of cryptographic keys and the creation of a profile for each trader. The profile reflects the trader's interest in products and possibly his/her authorized spending limits.

- **Auction Set Up:** This step sets up the auction protocol and rules such as item descriptions, auction type (e.g., open cry, sealed bid or Dutch), negotiated auction parameters (e.g., price, delivery dates, terms of payment), auction starting time and duration, and auction closing conditions.

- **Scheduling and Advertising:** In order to attract potential buyers, items in a given category (e.g., art or jewelry) are generally auctioned together on a regular schedule. Popular items are sometimes mixed with less popular items. Items to be sold in upcoming auctions and the dates of upcoming auctions are advertised.

- **Bidding:** The bidding step handles the collection of bids from potential buyers. It implements the bid control rules (e.g., minimum bid, bid increment, bid deposit). It also notifies auction participants when higher bids are received.
- **Bid Evaluation and Auction Closing:** This step implements the auction closing rules and notifies the winners and losers.
- **Trade Settlement:** This final step handles payments to sellers and the transfer of goods to buyers. If the seller is not the auctioneer, this final step also includes the payment of fees to the auctioneer and other agents.

2.2 Internet Auction Fraud

Criminals have discovered the Internet to be a highly profitable venue for conducting illicit business activities [7]. Organized crime groups are involved in numerous technology-enabled crimes, including Internet auction fraud [3].

Sakurai and Yokoo [18] have observed that anonymity is an important factor in perpetrating Internet fraud and that the existence of indivisible bids causes difficulty in matching supply and demand. This is because a buyer or seller can submit a false name bid by pretending to be a potential buyer or seller, thereby manipulating the balance of supply and demand. Chae, *et al.* [1] have confirmed these observations, concluding that online auction fraud is successful due to information asymmetry and anonymity.

Chua and Wareham [4] have listed some of the reasons for the proliferation of Internet auction fraud. The high degree of anonymity is at the top of the list; it is easy for dishonest users to evade prosecution. Second on the list is the low cost of entry and exit. Interestingly, these are precisely the reasons for the success of Internet auctions.

According to the 2008 Internet Crime Report [9], the median loss per Internet fraud complaint in the United States was $931 in 2008; the total loss was $264.6 million. In all, there were 275,284 Internet crime complaints – auction fraud, non-delivery of purchased goods, credit/debit card fraud, computer intrusions, spam and child pornography. However, Internet auction fraud was the most commonly reported offense, comprising 25.5% of all complaints and 16.3% of the total reported loss. The average median loss per auction fraud complaint was $610.

Table 1. Internet fraud taxonomy.

Seller as Fraudster	
Bid Shilling	Seller bids on his own items to drive up the price
Misrepresentation	Seller intentionally misrepresents an item
Fee Stacking	Seller adds hidden costs such as handling charges after the auction
Failure to Ship	Seller does not send the items to the buyers
Reproductions and Counterfeits	Seller advertises counterfeit items as the real thing
Triangulation Fencing	Stolen items are sold
Shell Auction	Seller sets up an auction solely to obtain bank account and credit card information
Buyer as Fraudster	
Bid Shielding	Two buyers collude – one makes a low bid, while the other makes an inflated bid; the higher bidder withdraws before the auction ends
Failure to Pay	Buyer does not pay for the items
Buy and Switch	Buyer refuses the items, but keeps the original items and returns inferior items
Loss or Damage Claims	Buyer claims the items are damaged, disposes of them and requests a refund

Gregg and Scott [8] discovered that Internet auction fraud takes various forms, such as delivering goods that are different, of low quality, without ancillary components, defective, damaged or black market items.

Morzy [15] describe other practices, including bid shielding and bid shilling. Bid shielding is the offering of an artificially high bid for an item to discourage other bidders from competing for the item. At the last moment, the "shielder" withdraws the high bid, enabling the second highest bidder, who is usually an accomplice, to win the auction. Bid shilling involves the use of a false bidder identity to drive up the price of an item on behalf of the seller.

Gregg and Scott [8] note that accumulation fraud is on the increase. In this type of fraud, a seller builds his reputation by selling large quantities of low-value merchandise over a long period of time. Having earned a good reputation, the seller offers expensive goods, but does not send the goods to buyers after receiving payment for them.

Chua and Wareham [4] created the auction fraud taxonomy presented in Table 1. According to Chua and Wareham, all the types of fraud listed are very damaging to Internet auction houses. They undermine user trust, which is disastrous for business.

Ku, *et al.* [12] note that while both buyers and sellers can be victims of fraud, a buyer is more easily targeted than a seller. They observed that 89% of seller-buyer pairs conducted just one transaction during the time period of their study; at most, there were four transactions between a seller-buyer pair. This means that the repeated transaction rate for the same seller-buyer pair is lower than 2%. If the transaction rate is much higher than 2%, then the transactions between the seller-buyer pair are suspect and could involve bid shilling or bid shielding.

Kobayashi and Ito [11] observed that many fraudsters tend to make honest deals during the early stages of their auction lives. However, they commit fraud soon after earning good reputations.

Ochaeta [16] also observed that fraudsters tend to establish good reputations prior to committing fraudulent acts. Therefore, the reputation building process of fraudsters is different from that of legitimate users. Specifically, fraudsters attempt to gain as much one-time profit as possible and as quickly as practicable. Consequently, fraudsters can be identified based on their reputation-building activities.

Fraudsters attempt to build their reputations by buying or selling numerous cheap items from sellers with good reputations. Additionally, they may buy or sell moderately priced or expensive items to accomplices. These buying and selling activities generally take place over a short period of time.

In order to build good reputations over a short period of time, most Internet auction fraudsters tend to sell large amounts of low-priced products. These sales take place at the beginning of their fraudulent auction lives. Also, fraudsters may attempt to bid for inexpensive items from sellers with good reputations. This is done to establish a favorable reputation by conducting many legitimate transactions.

3. Internet Auction Fraud in Hong Kong

We conducted a statistical analysis of 278 cases in Hong Kong to reveal the characteristics of Internet auction fraud related to the sale of counterfeit goods. The cases were the result of complaints lodged with the Hong Kong Customs and Excise Department. The following characteristics were observed in the analyzed cases:

- Fake goods are sold at unreasonably low prices, about 10% of the prices of legitimate goods.
- In about two-thirds of the cases (180 out of 278), the goods are sold within seven days of account creation.

- Fraudsters have multiple auction accounts that do not carry high trust values or reputation scores (8 out of 10 or higher).
- Fraudulent accounts are short lived (less than ten days) and fraudsters tend to switch to other auction accounts before the auction period expires.
- Many categories of goods (more than five) are sold (e.g., watches, mobile phones, footwear and sportswear).

4. Investigative Model

This section describes an investigative model for online auction fraud involving the sale of counterfeit goods. The model employs a Bayesian network to support the reasoning about evidentiary hypotheses.

4.1 Hypotheses and Evidentiary Traces

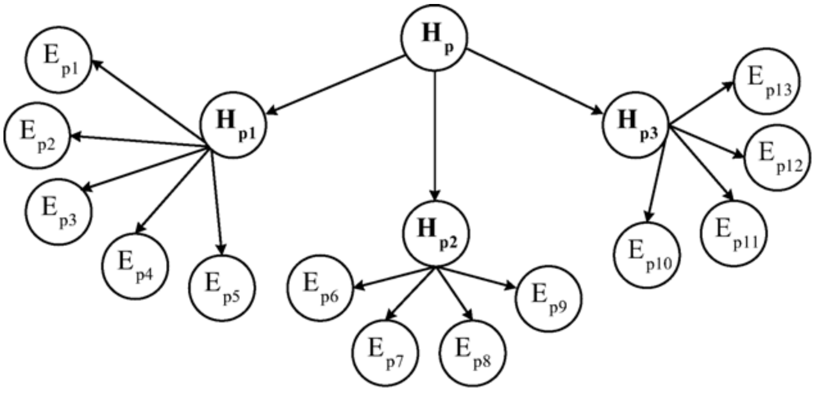
Digital evidence related to twenty prosecuted cases from the 278 complaints of selling counterfeit goods in Internet auctions was used to frame three sub-hypotheses about the actions taken by fraudsters. Because detailed judgments were not available for the prosecuted cases, digital forensic examiners who worked on the cases were interviewed to elicit the sub-hypotheses. The three sub-hypotheses are:

- Auction-related materials (e.g., images and item descriptions) were downloaded.
- The auction item (e.g., price of the item) was manipulated.
- The buyer and seller communicated (e.g., via email or instant messaging) about the counterfeit item.

These three sub-hypotheses substantiate the overall prosecution hypothesis that an online auction fraud crime was committed in the twenty prosecuted cases. The sub-hypotheses are supported by thirteen distinct evidentiary traces, which were obtained from the responsible digital forensic examiners. The various hypotheses and evidentiary traces are expressed using a Bayesian network model shown in Figure 1.

This investigative model does not of itself substantiate the entire prosecution case. The auctioned item has to be procured by the investigator and then be examined by the trademark owner to ascertain whether or not the item is counterfeit.

In order to evaluate the relevance of the digital evidential traces, a second simple Bayesian network model is created to express the defense



Hypotheses

H_p: **Seized computer was used as a transaction tool for the auction of the counterfeit item**
H_{p1}: **Uploading of auction material related to the counterfeit item was performed**
H_{p2}: **Manipulation of the corresponding auction item took place**
H_{p3}: **Communication between the seller and buyer about the counterfeit item occurred**

Evidence

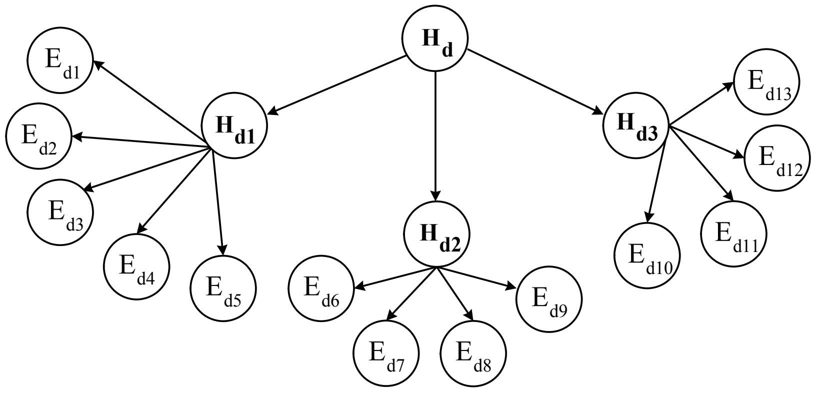
E_{p1}: **Information about the counterfeit item (e.g., image, description) was found on the seized computer**
E_{p2}: **Seller's account login record was retrieved from the auction site**
E_{p3}: **File metadata found on the seized computer matched the metadata found on the auction site**
E_{p4}: **IP address assigned to the seized computer matched the IP address used for data transfer**
E_{p5}: **Internet history/cache contents on the seized computer indicated the transfer of the counterfeit item**
E_{p6}: **Seller's account login record was retrieved from the auction site**
E_{p7}: **IP address assigned to the seized computer matched the IP address used for data transfer**
E_{p8}: **Editing of the auction item (e.g., price adjustment) occurred on the auction site**
E_{p9}: **Information about the auction item (e.g., image, description) was found on the seized computer**
E_{p10}: **Messages from the auction site related to the auction item were found on the seized computer**
E_{p11}: **Messages to/from the buyer related to the auction item were found on the seized computer**
E_{p12}: **Address book containing the covert investigator's email address was found on the seized computer**
E_{p13}: **IP address assigned to the seized computer matched the IP address used for email communication**

Figure 1. Bayesian network model for prosecution hypotheses and evidentiary traces.

viewpoint. Figure 2 presents the defense hypotheses and their associated evidentiary traces. Although the root hypotheses of the defense and prosecution models appear to be the same, they are, in fact, different because of the different supporting sub-hypotheses that express the defense and prosecution viewpoints. However, the same set of evidentiary traces is used in both models.

4.2 Evidence Evaluation

We use the likelihood ratio (LR) to evaluate the evidence in Internet auction fraud cases. LR is a general technique that can be applied to any scenario with decision uncertainty. In particular, it is very effective for quantifying the value or relevance of evidence [14]. The closer the LR



Hypotheses
H_d: Seized computer was used as a transaction tool for the auction of the counterfeit item
H_{d1}: Downloading of auction material related to the counterfeit item was performed
H_{d2}: Manipulation of the non-counterfeit auction item took place
H_{d3}: Communication between the seller and buyer about the non-counterfeit item occurred

Evidence
E_{d1}: Information about the counterfeit item (e.g., image, description) was found on the seized computer
E_{d2}: Seller's account login record was retrieved from the auction site
E_{d3}: File metadata found on the seized computer matched the metadata found on the auction site
E_{d4}: IP address assigned to the seized computer matched the IP address used for data transfer
E_{d5}: Internet history/cache contents on the seized computer indicated the transfer of the counterfeit item
E_{d6}: Seller's account login record was retrieved from the auction site
E_{d7}: IP address assigned to the seized computer matched the IP address used for data transfer
E_{d8}: Editing of the auction item (e.g., price adjustment) occurred on the auction site
E_{d9}: Information about the auction item (e.g., image, description) was found on the seized computer
E_{d10}: Messages from the auction site related to the auction item were found on the seized computer
E_{d11}: Messages to/from the buyer related to the auction item were found on the seized computer
E_{d12}: Address book containing the covert investigator's email address was found on the seized computer
E_{d13}: IP address assigned to the seized computer matched the IP address used for email communication

Figure 2. Bayesian network model for defense hypotheses and evidentiary traces.

value is to one, the less relevant is the evidence. Evett [6] generalized the LR approach to represent a situation where it is uncertain if the evidence is the result of the activities of a suspect. The general form proposed by Evett is:

$$LR = \frac{Pr(E|H_p)}{Pr(E|H_d)}$$

where E is the total digital evidence related to the crime, and H_p and H_d are the overall prosecution hypothesis and the overall defense hypothesis, respectively.

In our simple Bayesian network model, the existence of each individual trace of digital evidence does not imply the existence of any other traces. Since the evidentiary traces are mutually independent, their individual

Table 2. Conclusions drawn on LR values.

Likelihood Ratio	**Evidentiary Support**
1 to 10	Limited
10 to 100	Moderate
100 to 1,000	Moderately Strong
1,000 to 10,000	Strong
More than 10,000	Very Strong

probabilities can be multiplied together to determine the probability of E given a root hypothesis. The prior probability values of the individual evidentiary traces for the Internet auction fraud models (prosecution and defense) were obtained by surveying digital forensic examiners with the Hong Kong Customs and Excise Department, and are generally accepted values within this community of experts.

Evaluation of the Individual Sub-Hypotheses To evaluate the evidentiary relevance or LR values of the individual sub-hypotheses, it is necessary to set the individual sub-hypotheses to "Yes" separately and then multiply the prior probability values of their associated evidence. Thus, the LR value of evidence for hypothesis H_p against the evidence for hypothesis H_d $(Pr(E|H_p)/Pr(E|H_d))$ is given by:

$$\begin{aligned} & \frac{Pr(E_{p1}|H_{p1}) \times Pr(E_{p2}|H_{p1}) \times Pr(E_{p3}|H_{p1}) \times Pr(E_{p4}|H_{p1}) \times Pr(E_{p5}|H_{p1})}{Pr(E_{d1}|H_{d1}) \times Pr(E_{d2}|H_{d1}) \times Pr(E_{d3}|H_{d1}) \times Pr(E_{d4}|H_{d1}) \times Pr(E_{d5}|H_{d1})} \\ \approx\ & \frac{0.9 \times 0.75 \times 0.6 \times 0.75 \times 0.85}{0.9 \times 0.05 \times 0.6 \times 0.01 \times 0.01} \approx \frac{0.258}{0.0000027} \approx 95,600 \end{aligned}$$

Applying the interpretation adopted by the U.K. Forensic Science Service [10], an LR value of 95,600 indicates very strong support of the evidence for the prosecution's claim over the defense's claim. Table 2 illustrates the interpretation used by the Forensic Science Service.

Similarly, the LR values for H_{p2} against H_{d2} and for H_{p3} against H_{d3} are given by:

$$\begin{aligned} \frac{Pr(E|H_{p2})}{Pr(E|H_{d2})} &\approx \frac{0.247}{0.000319} \approx 774 \\ \frac{Pr(E|H_{p3})}{Pr(E|H_{d3})} &\approx \frac{0.190}{0.000938} \approx 203 \end{aligned}$$

The computed LR value indicates very strong evidentiary support for the prosecution's sub-hypothesis H_{p1}. On the other hand, the LR values indicate that the evidence supports the prosecution's sub-hypotheses H_{p2} and H_{p3} moderately strongly.

A limitation exists in the application of the LR approach to evaluate the evidentiary relevance of individual sub-hypotheses. In order to compute LR values, the corresponding sub-hypotheses should exist in the Bayesian network models expressing the prosecution and defense viewpoints. This requirement renders the LR approach inapplicable when the sub-hypotheses in two models do not correspond to each other (e.g., the number of sub-hypotheses in the defense Bayesian network model is larger than the number in the prosecution model).

However, under normal circumstances, the sub-hypotheses in both models will correspond because most of the sub-hypotheses in the defense model stem from the sub-hypotheses in the prosecution model. Evaluating the evidentiary relevance of individual sub-hypotheses can identify the strongest and weakest sub-hypotheses in the models. This enables digital forensic practitioners to identify the most significant and/or the most insignificant groups of evidence that are encompassed by the individual sub-hypotheses.

Evaluation of the Overall Hypotheses To compute $Pr(E|H_p)$, it is necessary to set the root hypothesis H_p of the prosecution Bayesian network to "Yes" and then multiply the resulting probability values of E_{p1} to E_{p13}. Similarly, to compute $Pr(E|H_d)$, it is necessary to set the root hypothesis H_d of the defense Bayesian network to "No" and multiply the resulting probability values of E_{d1} to E_{d13}. Specifically, we have:

$$Pr(E|H_p) \approx 0.000293; \quad Pr(E|H_d) \approx 0.00000000179$$

Hence,

$$LR = \frac{Pr(E|H_p)}{Pr(E|H_d)} \approx \frac{0.000293}{0.00000000179} \approx 164,000$$

The LR value of 164,000 indicates very strong evidentiary support for the prosecution's claim over the defense's claim.

5. Conclusions

The analysis of allegations of counterfeit goods at Internet auction sites provides interesting insights into fraudster behavior. Bayesian networks and likelihood ratio values offer a powerful mechanism for analyzing the relevance of evidence in such cases. If all the evidentiary traces are initially assumed to be present, the LR values computed from the prosecution and defense models can be used as criteria to determine whether or not it is worthwhile to proceed with the search for evidentiary traces. Specifically, if the LR value is relatively large (greater than 1,000) the search for the implied digital evidence should proceed. This would be followed by applying a cost-effective digital forensic investigation model [17] to identify the evidentiary traces and then applying the Bayesian network model [13] with the retrieved traces. On the other hand, if the LR value is found to be relatively small, the evidence does not strongly support the chosen hypotheses. Therefore, the prosecution should review its hypotheses and/or the implied evidentiary traces.

Acknowledgements

The authors wish to thank Dr. Jeroen Keppens of the Department of Computer Science, King's College London for his technical assistance.

References

[1] M. Chae, S. Shim, H. Cho and B. Lee, Empirical analysis of online auction fraud: Credit card phantom transactions, *Expert Systems with Applications*, vol. 37(4), pp. 2991–2999, 2010.

[2] China Internet Network Information Center, Statistical Survey Report on Internet Development in China (Abridged Edition), Beijing, China (www.cnnic.net.cn/uploadfiles/pdf/2008/8/15/145744.pdf), 2008.

[3] R. Choo, Organized crime groups in cyberspace: A typology, *Trends in Organized Crime*, vol. 11(3), pp. 270–295, 2008.

[4] C. Chua and J. Wareham, Self-regulation for online auctions: An analysis, *Proceedings of the Twenty-Third International Conference on Information Systems*, pp. 115–125, 2002.

[5] Data Center of the China Internet, The First Half of 2008 China Internet User Measurement Data IUI Index Report, Beijing, China, 2008.

[6] I. Evett, Establishing the evidential value of a small quantity of material found at a crime scene, *Journal of the Forensic Science Society*, vol. 33(2), pp. 83–86, 1993.

[7] S. Gajek and A. Sadeghi, A forensic framework for tracing phishers, *Proceedings of the Third International Conference on the Future of Identity in the Information Society*, pp. 19–33, 2008.

[8] D. Gregg and J. Scott, A typology of complaints about eBay sellers, *Communications of the ACM*, vol. 51(4), pp. 69–74, 2008.

[9] Internet Crime Complaint Center, 2008 Internet Crime Report, National White Collar Crime Center, Richmond, Virginia, 2008.

[10] J. Keppens, Towards qualitative approaches to Bayesian evidential reasoning, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, pp. 17–25, 2007.

[11] M. Kobayashi and T. Ito, A transactional relationship visualization system in Internet auctions, *Studies in Computational Intelligence*, vol. 110, pp. 87–99, 2008.

[12] Y. Ku, Y. Chen and C. Chiu, A proposed data mining approach for Internet auction fraud detection, *Proceedings of the Pacific Asia Workshop on Intelligence and Security Informatics*, pp. 238–243, 2007.

[13] M. Kwan, K. Chow, F. Law and P. Lai, Reasoning about evidence using Bayesian networks, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 275–289, 2008.

[14] D. Lucy, *Introduction to Statistics for Forensic Scientists*, Wiley, Chichester, United Kingdom, 2005.

[15] M. Morzy, New algorithms for mining the reputation of participants of online auctions, *Algorithmica*, vol. 52(1), pp. 95–112, 2008.

[16] K. Ochaeta, Fraud Detection for Internet Auctions: A Data Mining Approach, Ph.D. Thesis, College of Technology Management, National Tsing-Hua University, Hsinchu, Taiwan, 2008.

[17] R. Overill, M. Kwan, K. Chow, P. Lai and F. Law, A cost-effective model for digital forensic investigations, in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 231–240, 2009.

[18] Y. Sakurai and M. Yokoo, A false-name-proof double auction protocol for arbitrary evaluation values, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 329–336, 2003.

Chapter 10

DETECTING PONZI AND PYRAMID BUSINESS SCHEMES IN CHOREOGRAPHED WEB SERVICES

Murat Gunestas, Murad Mehmet and Duminda Wijesekera

Abstract Businesses are increasingly using web service choreographies to implement dynamic service invocations and content specific operations. These web service choreographies can be misused at multiple levels – by exploiting their technical capabilities and using them to design complex illegal business schemes such as Ponzi, pyramid and money laundering schemes. One of the main problems with the illegal schemes is that they are similar to legal multistage business schemes; their illegality is apparent only to a macroscopic observer. This paper describes some of these schemes and demonstrates how to obtain evidence pertaining to the schemes using cryptographically-secure local message repositories. The evidence gathered is of considerable value to financial fraud investigators, business arbiters, potential investors and judicial actors.

Keywords: Web services, choreographies, Ponzi schemes, pyramid schemes

1. Introduction

Businesses are increasingly invoking dynamic services and generating content-specific operations among choreographed web services, thereby creating service interdependencies between web services. These dynamic service interdependencies can be exploited to create new misuse activities. Some exploit the infrastructural dependencies of the services themselves while others use the infrastructural dependencies to create illegal business schemes. This paper focuses on detecting Ponzi and pyramid investment schemes created to defraud unsuspecting investors.

Illegal business schemes are difficult to detect because they are similar to legal business schemes at a microscopic level and are apparent only at the macroscopic level. Thus, they can elude local monitoring of web

Table 1. Pyramid scheme.

Level	Payments of $400 (#)		
1 ($100 × 3 = $300)	1 × #	1 × #	1 × #
2 ($30 × 9 = $270)	3 × #	3 × #	3 × #
3 ($30 × 27 = $810)	9 × #	9 × #	9 × #
4 ($30 × 81 = $2,430)	27 × #	27 × #	27 × #
...	...	...	...
21	10,460,353,203 × #		

transactions. Ponzi and pyramid schemes [13, 14] are difficult to differentiate from multilevel marketing schemes that either run their own businesses or invest in others. The basic dynamic of these two schemes is to rob Peter to pay Paul [16].

The Ponzi scheme is named after Charles Ponzi [16]. For many years, he collected money and promised returns within 90 days to investors who enrolled other investors in the scheme. Ponzi paid out the early investors using funds invested by late joiners. Other than running this scheme, Ponzi neither ran a business nor invested in other businesses and, consequently, neither incurred a profit nor a loss.

Numerous Internet-based Ponzi schemes are currently being investigated [9, 10]. In 2006 alone, 25,000 web sites suspected of running Ponzi schemes were shut down by the U.S. Securities and Exchange Commission [7].

A classic pyramid scheme, shown in Table 1, also uses the same principle. The orchestrator who originates the scheme promises top-level investors large returns on their investments, as do the recruited investors to their potential recruits. The example in Table 1 is organized as a four-level payment scheme with a span of three, i.e., only up to four levels of ancestral recruiters profit from investments, and each recruit at every level recruits three others. A Level 1 investor recruits three others and receives $100 per recruit. Each recruit is expected to recruit three others, thereby building a recruit tree. The Level 1 investor is paid $30 (not $100) per recruit at Levels 2, 3 and 4; and does not profit from investors beyond Level 4.

The orchestrator considers the recruiting activity to be complete when he receives $400 from an investor. Therefore, the orchestrator pays his recruiters $100 for the first parent or $30 for the three immediate ancestral recruiters, paying at most $190 on an investment of $400. Consequently, an investor makes $3,810 ($2,430 + $810 + $270 + $300) on his investment of $400 to his promoter if he successfully recruits three other investors and they successfully recruit three other investors all the

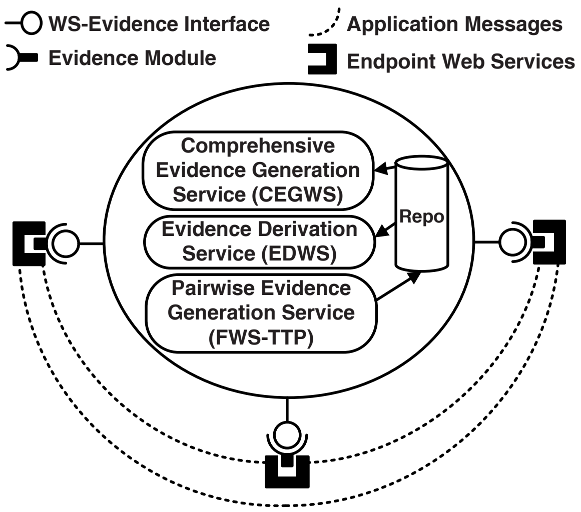

WS-Evidence Interface
Evidence Module
Application Messages
Endpoint Web Services
Comprehensive Evidence Generation Service (CEGWS)
Evidence Derivation Service (EDWS)
Pairwise Evidence Generation Service (FWS-TTP)
Repo

Figure 1. Evidence generation framework.

way down to Level 4. The scheme grows quickly, but is unsustainable in the long term and many promoters lose their investments because of the difficulty of attracting new recruits [14].

Financial institutions and their business partners are increasingly engaging service-oriented architectures such as dynamic brokering, creating dynamic service interdependencies that can be exploited to orchestrate illegal business practices. In order to detect such illegal schemes, it is necessary to have a comprehensive non-repudiable perspective of complex multi-party transaction models. A multi-party communication can arise in two possible ways among dynamic web services. In the first, a static communication pattern is specified and all participants follow this previously-known pattern. In the second, web services discover and transact with other services, dynamically creating choreographies that were unknown *a priori*. Consequently, in order to discover illegal activity, it is necessary to identify illegal business transactions using both types of choreographies.

2. Evidence Generation Framework

This section briefly reviews the evidence generation framework (EGF) of Gunestas, *et al.* [2], which is used to track business-level choreographic misuse. The EGF has three layers (Figure 1). The pairwise evidence generation service (Level 1) generates evidence for pairwise interactions between web services. The evidence derivation service (Layer 2) derives facts from available pairwise evidence in order to refute or justify claims of agreement violations between communicating partner services. The comprehensive evidence generation service (Layer 3) generates instances of requested choreographies from Layer 2 and Layer 3 data.

The EGF provides online evidence generation and management capabilities to other web services as a web service itself. In order to use the EGF, other web services (called member services) should integrate the EGF with themselves using a centralized service access point. Thereafter, the EGF acts as a trusted third party. As a service, the EGF receives and retains service requests and responds in a cryptographically-secure manner, retains the correspondence in secure repositories, and provides the correspondence for dispute resolution and forensic investigations. The EGF provides "evidence adapters" for all requests.

Gunestas, *et al.* [2] have constructed a prototype implementation of Layer 1 and several protocols based on one-way and request-response message exchange patterns. Currently, the EGF provides evidence for non-repudiation, fairness and timeliness using digital signatures to provide proofs of receipt and delivery, to link a message to its creator/sender and to provide message integrity. For accountability, the EGF uses fair non-repudiation mechanisms that utilize trusted third parties (TTPs). Some fair exchange protocols (e.g., [6]) do not use TTPs because they assume that the participants have prior knowledge of the message contents; but they are not used in the EGF because web services may not always know the expected message content. Timeliness is required because of the time-sensitive nature of most business transactions. Evidence records are based on the time observed at TTPs. EGF servers gather pairwise transactional evidence that flows between sender and receiver web services, employing inline TTPs that use the Simple Evidence Layer Protocol (SELP) or offline TTPs that use the Optimistic Evidence Layer Protocol (OELP) [4]. SELP and OELP are used by end-points to obtain non-repudiable evidence by engaging a specific message format and digital signatures.

3. Evidence of Observed Interactions

Web services use many kinds of messages (e.g., one-way messages and request-response messages) in order to choreograph business processes between themselves that correspond to the four proposed WSDL operation types (in-only, out-only, in-out and out-in). An external observer who is ignorant of the business processes can observe only the one-way and request-response message exchange patterns that are formalized in Definition 1.

Definition 1 (Message): A web services message consists of three components: (i) mandatory fields corresponding to the sender, receiver and time, where the first two are URLs and the last is chosen from a set T; (ii) optional fields corresponding to a finite set of attributes from a set A; and (iii) message content consisting of strings from an alphanumeric set C.

In this paper, the symbol $|$ denotes string concatenation and $encA(r)$ denotes the string obtained by encrypting r using A's key. Furthermore, if m is a message, then $m.a$ denotes the value of its attribute a. For example, $m.time$ is the value of the timestamp of m.

Definition 1 establishes the notation for describing the messages used to extract knowledge about externally-observable facts pertaining to choreographies. Because different choreographic specifications may select different labels for their identifier fields, in order to address naming convention problems, we use XPATH expressions to specify ID values. Furthermore, because any fabricator can produce messages, we rely on cryptographically-secure messages to ascertain reliable evidence. The messages are collected to derive "evidence objectives" – claims that are to be substantiated or refuted using the collected evidence, e.g., message origin, message properties or the intended recipient. This evidence is generated from cryptographically-secure messages. However, certain objectives such as evidence of delivery and evidence of non-availability may require messages to be signed by a TTP.

Definition 2 (Primitive Evidence Objectives): The three primitive evidence objectives are: (i) Evidence of Origin: message m with origin A and content $r|sigA(r)$ from A to B is said to provide evidence of origin; (ii) Evidence of Delivery: message m with content $ack|sigTTP(ack|m)$, where TTP is a trusted third party or content $ack|sigB(ack|m)$ and B is $m.recipient$ is said to provide evidence of delivery; and (iii) Message Evidence: evidence of a message m is a pair (m_1, m_2), where m_1 is the evidence of origin and m_2 is the evidence of delivery of m.

According to Definition 2, cryptographic evidence is required from a web service of a trusted third party for claims of origin and delivery. Interested readers are referred to [3, 4] for details about how the evidence of origin and evidence of delivery can be collected at TTPs using WS-Evidence messages that are generated via non-repudiation protocols.

The message evidence (ME) is stored in the form of log records in the EGF as described in [3]. Because log records may contain large volumes of data, message evidence indices (MEIs) are used to refer to messages. Table 2 shows a sample index table, where the first column is an index for stored packets that have the attributes of time, sender, message string

Table 2. Sample MEI table.

ID	Time	Sender	Receiver	Msg	Content
63..	21	A	B	r	<..invID..>
67..	22	B	C	m	“..”
68..	23	C	B	k	<..payID..>

and content. The first reference in the table is to a message sent by A to B with content <invID>.

4. Evidence of Global Misuse

Mining choreographies that are created due to message content from external observations require linkage parameters, which can be derived from externally-invisible message content; this renders them not very helpful to external monitors and auditors. One opportunity to obtain this information is when a victim makes a complaint. Thus, the following method can be used to detect a Ponzi-like scheme:

- Accept a victim complaint.
- Examine the content of specimen records involving promotion or investment messages provided by the victim.
- Determine the parameters in the evidence that can be linked.
- Detect choreographies and design them in the dynamics of the algorithms.
- Create the algorithms.
- Run the algorithms in the appropriate order, possibly running more than one algorithm when required.
- Collect a comprehensive set of evidence to determine if the scheme is illegal and its effect on the network.
- Broadcast an alert to current and potential victims.

The method described above works for hierarchical schemes. Different methods would be applied to other schemes. We apply our heuristic method to specify several algorithms for Ponzi-like schemes.

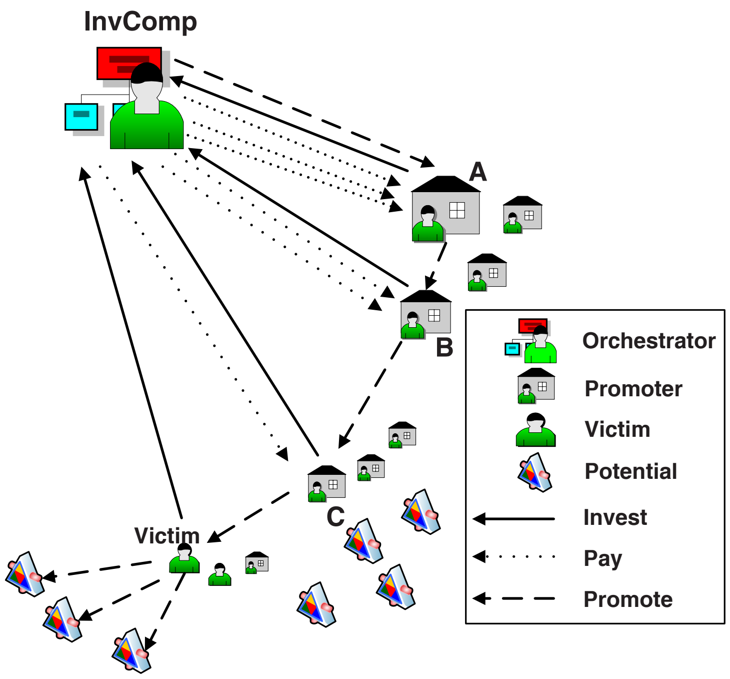


Figure 2. Ponzi-like recruits over web services.

4.1 Ponzi Schemes over Web Services

Typical Ponzi-like schemes have three types of actors: a malicious investment service acting as the orchestrator and investor services acting as promoters or victims (depending upon their investments and return rates). Figure 2 illustrates how these actors collaborate to spread the financial scheme over a network of web services. The investment company, InvComp (Orchestrator), promotes A (Promoter) to recruit B into its scheme by promising a quick return on investment and encourages B to promote the scheme to other potential investors. This promoting activity (using promote messages) may not necessarily be observed in the records because promoters may choose other means to convince investors. If A invests (using an invest message) in InvComp, then we say that A has been recruited. A now starts promoting InvComp to other investors in order to get a quick return on his investment and in the process recruits B. We recognize that B has been promoted by A because of the reference value in the content field of the invest message sent by B to InvComp. In accordance with the return policy of the scheme, InvComp makes a payment to A (pay message).

Table 3. MEI tuples featuring a misuse scheme.

ID	**Time**	**Sender**	**Receiver**	**Msg**	**Content**
...					
	45	B	InvComp	invest	Promoter=A
	55	InvComp	A	pay	150
...					
	67	C	InvComp	invest	Promoter=B
	76	InvComp	B	pay	150
	78	InvComp	A	pay	30
...					
	87	Victim	InvComp	invest	Promoter=C
	89	InvComp	C	pay	150
	92	InvComp	B	pay	30
	104	InvComp	A	pay	30

The choreographies between the investment company, recruiters and recruits spread in an investor web service network. When a recruit cannot attract enough investors, he loses his money (Victim). Figure 2 shows the complete and incomplete recruit choreographies.

We do not assume that we know the global scheme when mining. Instead, we assume that either a promoter web service identity or an invest message is submitted to a law enforcement agency by a victim. Following the heuristic method presented above, it is possible to find invest, pay and promote messages that contain attributes that refer to each other, demonstrating the collaboration in a pervasive manner.

4.2 Pattern Discovery

This section shows how to discover the patterns that help create comprehensive evidence of illegal business schemes. Following the heuristic method described above, we assume that a victim brings an invest message that contains a promoter web service. An algorithm executed on the evidence repository could show that the promoter web service has received pay messages from other web services and has sent an invest message to the same web service of the alleged investment company. Table 3 shows sample records corresponding to the misuse scheme.

The records in Table 3 show the pattern that keeps the fraudulent activity alive – invest messages are linked by sender header fields and promoter content fields. In other words, every promoter web service in an invest message is paid right after the invest message. The message

Table 4. Ponzi scheme with fanout 1 and depth 1.

I	invest;p pay where invest.sender=A and invest.reciever=B and pay.sender=B and pay.reciever=C and invest.prometer=pay.reciever
II	invest1;p invest2 where invest1.sender=C and invest1.receiver=B and invest2.sender=A and invest2.reciever=B and invest2.promoter= invest1.sender

evidence can be correlated to conclude that the promoter, victim and orchestrator may be involved in a hidden recruit choreography.

In cases where the promoting activity does not involve a web service message, the message or choreography patterns can be included as part of the recruiting activity. We call this content-based choreography "recruit." Pattern I in Table 4 is the signature for the "rob Peter to pay Paul" activity. Pattern II is used as the link between the recruiter and recruit, enabling the mining of recruit paths to create recruit trees from MEI records. For simplicity, we define patterns succinctly; however, more complex patterns may include pay messages, which increases the complexity of queries.

Table 3 shows that investors assume a promoter role within a subsequent recruit choreography pattern, thus creating a recursive investment scheme. For example, note that Investor B sends an invest message to InvComp at Time 45. At Time 67, Investor C makes a reference to B through his investment message and B receives a payment from InvComp some time later (pay message at Time 76). The same choreography can be observed between C and other subsequent investors, revealing that they recruited other investors shown in later records. Consequently, in order to detect such a scheme, it is necessary to recognize the recursive investment and payback schemes. The recursive scheme creates a recruit tree that joins a recruiter to all his recruits. By traversing a path in a recruit tree from a chosen (victim) node to the root of the tree (say recruit paths), it is possible to identify the orchestrator. The path Victim-C-B-A in Figure 2 is such a recruit path.

In this paper, we use the following notation to specify recruit trees formally. Let k be an integer that denotes the fanout of the recruit tree. Then, all finite sequences of $\{0, \ldots, k-1\}$ are used as identifiers for web service nodes. We use the notation $2^{k<w}$ to denote the set of finite subsequences of $\{0, \ldots, k-1\}$. For example, all binary sequences can be used to index trees with fanout 2, where the left child of node x_p is x_{p0} and the right child is x_{p1}, where p is a finite sequence of integers $\{0,1\}$, i.e. $2^{<w}$. The notation p<q denotes that p is a subsequence of

q where $p, q \in k^{<w}$. The length of $p \in k^{<w}$ is denoted by |p|. Now suppose $p \in k^{<w}$ where $|p|= m$ and $p =<p_o, \ldots, p_{m-1}>$. Then, the i^{th} ancestors of p for $i \geq 1$ are given by ancestor(i)=$<p_o, \ldots, p_{m-i}>$. Finally, ϕ denotes the empty string in $k^{<w}$.

Definition 3 (Recruit trees of fanout k and depth m): Suppose I is an investment company web service. Inductively define active(n) for every integer n as follows:
(i) Active(0)={m} where m is defined as a message where [m.sender=p_o, m.receiver=I].
(ii) Suppose active(n) has been defined and $p \in k^{<w}$ with $|p|= n$, then for each $i \in \{0, \ldots, k-1\}$ define [active($p\hat{}i$)=$msg_{;p}$PayBack($p\hat{}i$)] where msg satisfies [msg.sender=$P_{p\hat{}i}$, msg.reciever=I and msg.content= invest] and PayBack($p\hat{}i$) satisfies PayBack($p\hat{}i$)=$msg_1 \cap_p \ldots \cap_p msg_m$ where every msg_i is of the form [msg_i.content=pay, msg_i.sender=I, msg_i.receiver=ancestor(p,i)] for $i = l$.
(iii) Let active($n+1$)=active($p\hat{}0$) $\cup_p \ldots \cup_p$ active($p\hat{}(n-1)$).
(iv) Define a recruit tree to be active* = LFP(f, m, E) where function f is defined in items (i) and (ii); the message m is defined in item (i); and the set of message equations E is defined in items (i) and (ii).

We denote the class of Ponzi schemes of fanout k and depth l and attribute equations E as Ponzi(k, l, E), where E is the collection of equations in Definition 3. Definition 3 provides a generic definition for Ponzi-like schemes where the number of recruits employed by any recruiter is limited to an integer k and the number of ancestors deriving a payback from the recruitment is at most an integer l.

The web service nodes are numbered by strings from $\{0, \ldots, k-1\}$, resulting in trees where every node has at most n levels. Thus, the parameter p in Definition 3 denotes a path with $|p|$ elements in such a tree. Step 0 with the empty string ϕ represents the recruiter in item (i) in Definition 3. Item (ii) assumes that the tree is defined up to a path p of length n and finds its next level. This step is the sequential composition of two steps. First, P_p sends messages to each of its children to invest. Next, each of these children invest in I, followed by I paying the ancestors of these children. The ancestors who are paid back are limited to at most l generations. Item (iii) in Definition 3 collects all possible paths that extend the tree to the next level $n+1$. Item (iv) collects all the sub-trees with depth $n+1$.

5. Detecting Global Misuse

This section discusses how to mine global misuse instances of given patterns from log records of observed web transactions using Stream-

Algorithm 1 Detecting Recruits of Ponzi Schemes

INPUT: MEI tuples
OUTPUT: Ponzi-like recruit MEI pairs
DESCRIPTION: Glides over MEIs using a window size of 3 to detect Pattern I along with the predicates specified in the WHERE phrase

```
 1  CreateInputStream MEI ($MEI schema) ;
 2  CreateOutputStream PonziDetectOut ;
 3  CreateStream InvestFilterOut ;
 4  CreateStream PayFilterOut ;
 5  SELECT * FROM MEI
 6  WHERE msg==“invest” INTO InvestFilterOut
 7  WHERE msg==“pay” INTO PayFilterOut
 8  SELECT “Ponzi-like recruit” AS detected, invest.time AS investTime,
    pay.time AS payTime, pay.receiver AS recruiter, invest.sender AS recruit
 9  FROM PATTERN (InvestFilterOut AS invest THEN PayFilterOut AS pay)
10  WITHIN 3 (days) ON time
11  WHERE invest.receiver==pay.sender AND regexmatch
    (“.*”+“promoter=”+pay.receiver+“.*”, invest.content)
12  INTO PonziDetectOut ;
```

SQL [12] and a StreamBase platform [11]. StreamSQL is an event pattern language that can be used to define queries over streams of data. StreamBase is an event processing platform that can run StreamSQL queries over input source file or database and produce outputs.

StreamSQL has several commands. CREATE INPUT STREAM creates data streams from a named file that is pre-configured according to a known schema. CREATE OUTPUT STREAM creates an output stream that is pre-configured according to a schema. A PATTERN phrase defines the search criteria from multiple input streams. A WITHIN phrase creates the maximum size of a window that moves along a collection of aligned streams when searching for a pattern.

Algorithm 1, which is based on StreamSQL, discovers Pattern I defined in Table 4. The algorithm accepts MEI records in ascending order of timestamps. The algorithm processes the pattern by filtering the records into two groups, invest and pay, using the predicates defined in Lines 6 and 7. This enables the pattern to employ the appropriate template (THEN phrase) in Line 9, i.e., invest messages are expected before pay messages. The predicates defined in Line 11 stipulate that the receiver of the invest message should be the sender of the following pay message; and the promoter value in the content of the invest message should be the receiver of the following pay message. The window size is set to 3 in Line 10. The SELECT phrase gathers the re-

Algorithm 2 Enhancing Ponzi Detection

INPUT: PonziDetectOut from detectRecruits
OUTPUT: Ponzi alerts
DESCRIPTION: Counts detected Ponzi-like recruits using a window size of 6 as the minimum support. Emits Ponzi alerts when the minimum support is reached

```
SELECT "Ponzi Alerts," count() AS minSup
FROM PonziDetectOut [SIZE 6 TUPLES]
INTO PonziAlerts ;
```

quired information about the detected pattern and emits the result to the PonziDetectOut table.

Detecting a few Ponzi-like recruits may not be sufficient to declare that a Ponzi scheme exists. To increase the confidence, a minimum support value is defined as a threshold; and an alert is sent only when the threshold is exceeded. Algorithm 2 can be employed to strengthen Algorithm 1 by incorporating a predefined minimum support value.

Algorithm 2 sends an alert when at least six Ponzi-like recruits are detected in the output of Algorithm 1. Note that the addition of Algorithm 2 to Algorithm 1 decreases the number of false positives.

6. Generating Comprehensive Evidence

This section discusses how to detect the orchestrator or the earliest known recruiter of a Ponzi scheme by tracing a potential recruit path to its root. Next, all possible paths that originate at the detected recruiter are examined to identify others who have invested in the Ponzi scheme. We begin by defining a choreography.

Suppose that the complaint includes an invest message msg and that the investment company I is used by the participants in the scheme. We define an ancestorChain(n) as:

(i) ancestorChain(0) = msg

(ii) ancestorChain($n+1$) = $[msg_{l;p}\ (msg_{1;p}\ k_1) \cup_p\ \ldots \cup_p$ $(msg_{l-1;p}\ k_{l-1})\ _{;p}$ ancestor(ancestorChain(n),n)$]$

satisfying the equations E:

msg_1.content=pay and k_1.content=invest, $\ldots$,
msg_{l-1}.content=pay and k_{l-1}.content=invest
(msg_l.time $<$ msg_l.time $<$ k_1.time), $\ldots$,
(msg_1.time $<$ msg_{l-1}.time $>$ k_{l-1}.time) and msg_l.reciever=I.

Also, we define Earliest(msg) as LFP(f, msg, E).

As a special case, we show how to compute the ancestor chain corresponding to Pattern II in Table 4 using StreamSQL in Algorithm 3. Given a recruiter, the algorithm traces ancestor recruiters to find the

Algorithm 3 Computing the Orchestrator

INPUT: Promoter $P, MEI tuples
OUTPUT: Ancestor chain of promoters as RecruitPathOut
DESCRIPTION: Given the promoter, traces back the MEI records and finds the path and the distance to/from the orchestrator using Pattern II

```
 1 CreateInputStream MEI ($MEI schema) ;
 2 CreateOutputStream RecruitPathOut(
   $MEI schema, newPromoter String) ;
 3 CreateStream LocalStream ;
 4 DECLARE pointerPromoter String DEFAULT $ PUPDATE FROM
 5 SELECT newPromoter AS pointerPromoter FROM RecruitPathOut ;
 6 SELECT * FROM MEI
 7 WHERE msg==“invest” AND receiver==“O”
   AND sender==pointerPromoter
 8 INTO LocalStream
 9 SELECT time, sender, receiver, msg, content,
    GetXPATHValue(content,“../promoter/”) AS newPromoter
10 FROM LocalStream
11 INTO RecruitPathOut ;
```

orchestrator. It identifies when the scheme began by traversing records in descending order of timestamps looking for the senders of invest messages. This is done by declaring the dynamic variable pointerPromoter in Line 4 to which the suspected promoter is passed by default (see DEFAULT) as the orchestrator. Each time the output emits a hop (that meets the criterion in Line 7: invest message sent to the orchestrator) by the node to the receiver, the promoter value is extracted from the content of the invest message using a XPATH function (SELECT phrase in Line 9). It is then written to the output stream (Line 2) and assigned to the dynamic variable pointerPromoter in Line 5. The newly-assigned value is used as the predicate in Line 7 for locating the next message if it matches the sender value.

After the orchestrator or earliest recruits have been identified, a trace-forward algorithm is used to generate the evidence. Since the algorithm above locates one of the oldest message records for the trace-forward algorithm, it helps gather the most comprehensive evidence of the scheme.

Algorithm 4 compiles the evidence using Pattern II in Table 4. The algorithm accepts MEI tuples. The first SELECT phrase in Line 4 is a filter with predicates dealing with invest messages that are sent to the suspected orchestrator “O.” Pattern II is defined after the PATTERN phrase. The PATTERN phrase duplicates the invest MEIs so that it can apply the appropriate template (THEN phrase) and predicates (WHERE phrase in Line 8) between messages. The algorithm uses

Algorithm 4 Generate Recruit Tree

```
INPUT: MEI tuples
OUTPUT: RecruitsOut table leading to recruiter->recruit tree structure
DESCRIPTION: Tracing forward the MEIs, outputs an appropriate table
representing a tree-view of the scheme using Pattern II

CreateInputStream MEI ($MEI schema) ;
CreateOutputStream RecruitsOut ;
CreateStream InvestFilterOut ;
SELECT * FROM MEI
WHERE msg==“invest” AND receiver==“O” INTO InvestFilterOut ;
SELECT recruitee.time AS recruitTime, recruiter.sender
    AS recruiter, recruitee.sender AS recruit
FROM PATTERN (InvestFilterOut AS recruiter THEN
    InvestFilterOut AS recruitee) WITHIN 6 (days) ON time
WHERE recruiter.sender==
    GetXPATHValue(recruitee.content,“../promoter/”)
INTO RecruitsOut ;
```

a window of size 6, limiting it to finding patterns within the specified period. Queries with narrower windows may cause the algorithm to miss more correlations than queries with wider windows.

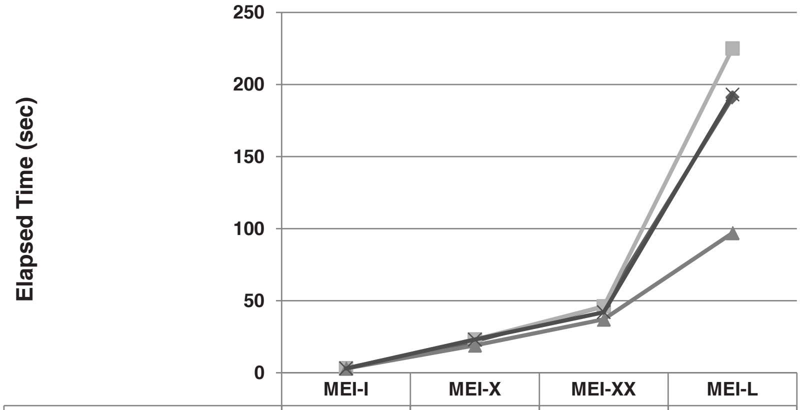
7. Experimental Validation

The three-layered evidence generation framework provides evidence of web service misuse. The services in the bottom and middle layers have been studied by others [1, 5, 8]. Therefore, we validate the top layer of the framework for which have introduced novel queries.

Table 5. Query-pattern mapping.

Query Name	**Pattern**	**Pattern Name**
DetectRecruits	invest;p pay	invest-pay
ClimbRecruitPath	invest1;p invest2	invest-invest
GenerateRecruitTree	invest1;p invest2	invest-invest

Table 5 summarizes the queries and their associated patterns along with the names we have used throughout this paper. In order to test their accuracy and performance rates, we generated synthetic data and used a special simulation platform described below.



	MEI-I	MEI-X	MEI-XX	MEI-L
◆ GenerateCHOR-Investing	3	22	46	191
■ DetectRecruits	3	23	46	225
▲ ClimbRecruitPath	3	19	37	97
✕ GenerateRecruitTree	3	23	42	193

Figure 3. Capacity values of test data.

7.1 Data Characteristics

To our knowledge, our illegal business scheme is novel and we were unable to obtain real data featuring the scheme. Consequently, we generated synthetic data in the MEI format called MEI-I that conforms with the Ponzi/pyramid schemes described above. MEI-I contains a total of 193,827 records (one record/second) over a period of three days (January 19, 2006 to January 21, 2006). The MEI-I file has a comma separated values (CSV) format. We focused on the capacity rate of data that embodies the illegal business scheme in its records.

To evaluate the performance, we created three more data sets using our seed data. The MEI-X data set contains the same Ponzi malicious activity, but is ten times larger than MEI-I in terms of the number of records. MEI-XX and MEI-L are two other sets that are 20 and 50 times larger than MEI-I, respectively (Figure 3).

7.2 Test Environment

As mentioned above, we defined use/misuse patterns using StreamSQL and employed the StreamBase platform to detect the patterns. We used the specially-generated data described above in the MEI structure. The feed simulation platform of StreamBase was adjusted to accept this data in the form of CSV files. The platform empowers users to run StreamSQLs over any user-defined file satisfying the data schema ex-

Table 6. Test environment

Hardware	Software
CPU: Intel Core2 T7400 2.16 GHz,	OS: Windows XP SP2
4 MB L2 Cache, 667 MHz FSB	JVM : SUN JDK 1.5.0.15
Physical Memory: 2 GB, 995 MHz	StreamBase Studio: Version 6.4
Hard Disk: 250 GB, 7200 rpm	Max Heap Size: 1024 MB

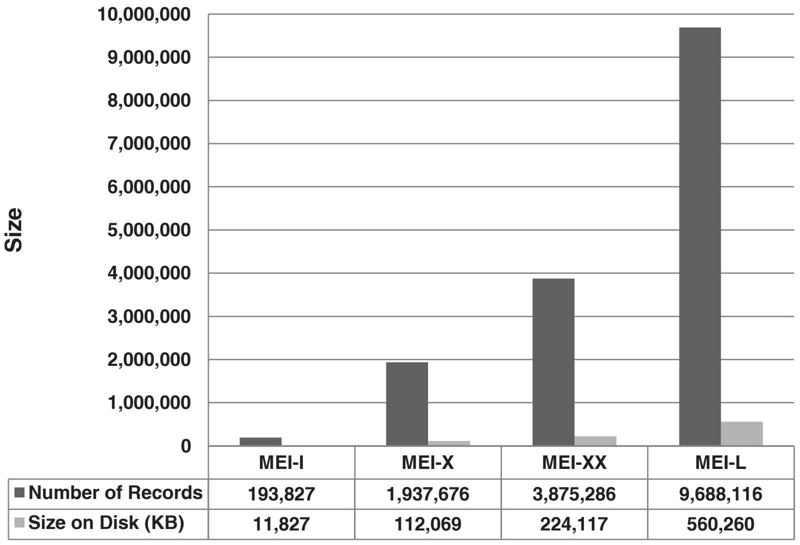
pected by the algorithm. The platform also provides observable outputs of algorithm execution. Using the StreamBase Manager, it is also possible to observe CPU and memory usage during algorithm execution. Although StreamBase encourages the use of enterprise servers for benchmarking and improved performance, we observed that the feed simulation platform was adequate to test our queries over vast amounts of data with a reasonable resource allocation rate. Table 6 lists the hardware and software components of the test environment.

7.3 Test Results

The performance of the algorithms was tested by executing three major queries (Table 5) over data sets of different sizes (Figure 4). As mentioned earlier, we observed that maximum proximity rates give the best decisions in the tests. Therefore, we built our performance tests using these rates in the algorithms.

Figure 8 presents the test results. Note that queries for detecting recruits and generating recruit trees take more time than queries for locating the orchestrator. The algorithm execution times for detecting Ponzi-like recruits are greater than the execution times for the other algorithms over the largest data set MEI-L. The recruit path climbing algorithm, which traverses records in the backward direction, exhibits the best performance and there is no need to use a larger window size.

We used the StreamBase Studio Feed Simulation platform in our validation tests. Other higher performance platforms are available, but we obtained reasonable performance despite using an integrated development environment instead of an enterprise environment. We also observed that the queries yield results with reasonable accuracy for the synthetic data sets. These results were obtained because we determined a reasonable window size for window-based queries for each data set. Also, we converged the evidence outcomes by tuning the time, property and key-based patterns associated with the queries.



	MEI-I	MEI-X	MEI-XX	MEI-L
■ Number of Records	193,827	1,937,676	3,875,286	9,688,116
■ Size on Disk (KB)	11,827	112,069	224,117	560,260

Figure 4. Performance test results.

8. Conclusions

Web service choreographies can be used as the foundation for detecting illegal business schemes. Our implemented system engages StreamSQL to define the associated use/misuse patterns and the StreamBase platform to detect the patterns in large streams of data. Although our choreographies only specify Ponzi-like schemes, the method can be used to specify (and detect) other illegal business activities [15] that can be mined from financial transaction repositories. Our future work will attempt to develop an online warning system that detects business schemes that appear legal from a microscopic view, but are illegal from a macroscopic perspective.

References

[1] M. Bilal, J. Thomas, M. Thomas and S. Abraham, Fair BPEL processes transaction using non-repudiation protocols, *Proceedings of the IEEE International Conference on Services Computing*, vol. 1, pp. 337–342, 2005.

[2] M. Gunestas, D. Wijesekera and A. Elkhodary, An evidence generation model for web services, *Proceedings of the IEEE International Conference on System of Systems Engineering*, pp. 1–6, 2009.

[3] M. Gunestas, D. Wijesekera and A. Singhal, Forensic web services, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 163–176, 2008.

[4] A. Herzberg and I. Yoffe, The Delivery and Evidence Layer, Cryptology ePrint Archive Report 2007/139 (eprint.iacr.org/2007/139 .pdf), 2007.

[5] A. Keller and H. Ludwig, The WSLA framework: Specifying and monitoring service level agreements for web services, *Journal of Network and Systems Management*, vol. 11(1), pp. 57–81, 2003.

[6] S. Kremer, O. Markowitch and J. Zhou, An intensive survey of non-repudiation protocols, *Computer Communications*, vol. 25(17), pp. 1606–1621, 2002.

[7] Los Angeles Times, Internet pyramid scheme alleged, September 28, 2006.

[8] A. Sahai, V. Machiraju, M. Sayal, A. van Moorsel and F. Casati, Automated SLA monitoring for web services, *Proceedings of the Thirteenth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, pp. 28–41, 2002.

[9] SavvySugar, Help! My friend is part of a pyramid scheme, *MainStreet*, June 23, 2009.

[10] R. Stewart, South Africa investigates an alleged Ponzi scheme, *The Wall Street Journal*, June 17, 2009.

[11] StreamBase Systems, StreamBase Products, Lexington, Massachusetts (www.streambase.com/products-home.htm).

[12] StreamBase Systems, StreamSQL, Lexington, Massachusetts (www .streambase.com/products-streamsql.htm).

[13] U.S. Security and Exchange Commission, Ponzi schemes, Washington, DC (www.sec.gov/answers/ponzi.htm).

[14] D. Valentine, Pyramid schemes, presented at the *International Monetary Fund's Seminar on Current Legal Issues Affecting Central Banks* (www.ftc.gov/speeches/other/dvimf16.shtm), 1998.

[15] C. Westphal, *Data Mining for Intelligence, Fraud and Criminal Detection: Advanced Analytics and Information Sharing Technologies*, CRC Press, Boca Raton, Florida, 2009.

[16] M. Zuckoff, *Ponzi's Scheme: The True Story of a Financial Legend*, Random House, New York, 2005.

Chapter 11

IDENTIFYING FIRST SEEDERS IN FOXY PEER-TO-PEER NETWORKS

Ricci Ieong, Pierre Lai, Kam-Pui Chow, Michael Kwan and Frank Law

Abstract This paper describes a new approach for identifying first seeders in illegal file sharing investigations involving Foxy, one of the most popular Chinese peer-to-peer networks. In identifying first seeders, the approach focuses on determining the slow-rising period of the cumulative seeder curve instead of merely measuring the number of seeders. The relationships between file popularity, number of packets and the maximum upload limit during the time that the first seeder is connected to the network are also analyzed. These relationships are used to specify rules that investigators can use to determine if an identified seeder is, in fact, the first seeder.

Keywords: Peer-to-peer network forensics, Foxy network, initial seeder

1. Introduction

The Foxy peer-to-peer (P2P) network is very popular in Chinese markets such as Hong Kong and Taiwan. It has approximately over 500,000 Chinese language users at any given time, including users from Asia, Australia, Europe and America. Foxy rapidly disseminates images, music and television programs throughout the global Chinese community. However, Foxy has also been used to distribute racy photographs, pornographic movies and sensitive documents. Once a file is shared on the network, it is practically impossible to completely remove the file, even if the owner wishes to control or block its distribution.

The identification of the first seeder is always the ultimate goal of a P2P file sharing investigation. One approach for verifying if an identified seeder is the first seeder is to determine if the seeder was connected during the slow-rising period of the cumulative seeder curve. However, it is difficult to identify the slow-rising period in a precise manner.

This paper describes an alternative approach to identify the slow-rising period instead of measuring the number of seeders. The approach is based on the observation that the file download duration fluctuates but reduces during the slow-rising period and becomes steady throughout the rapid-rising period; after this time, it increases again when peers leave the network. The paper also examines and clarifies the relationships between file popularity, number of packets and the maximum upload limit during the time that the first seeder was connected. These relationships are used to derive rules that help determine the first seeder in a P2P network.

2. Background

In the Foxy network, file contents are distributed from source uploaders or seeders. A seeder can either be a leaf node or hub node. Leaf nodes are ordinary nodes that contribute the majority of files shared in the Foxy network. Leaf nodes link to hub nodes, which act as "big brothers" who regulate and distribute relevant queries to the peers.

A node searching for a file sends the query to a hub node, which passes the query to the connected nodes. When a connected node indicates that it has the file available for download, the requesting node proceeds to download the file from the node, which becomes the seeder of the requested file. If the seeder is the first node in the Foxy network to upload a particular file, it is known as the first seeder of the file.

The first node that indicates the availability of a file for download could be the first seeder. The propagation of a file can be restricted by identifying and dealing with the first (and early) seeders before the file is widely disseminated. Moreover, the identification of the first seeder is a key goal in investigations involving the illegal distribution of files in P2P networks [1, 2, 5, 6]. This is because only the initial seeders of a file can be prosecuted for their intention to distribute the file under the Hong Kong legal system.

2.1 Identifying Initial Seeders

The initial seeder or first seeder is the seeder or seeders who initiate the distribution of a file in a P2P network. Peers download the file from this seeder. Observations of the download scenario from the beginning – before the file is distributed on the network – should make it easy to identify the first seeder. However, aside from the first uploader of the file, no one can know exactly when the file began to be distributed. The identification of first seeder is not a trivial task.

In some P2P networks (e.g., BitTorrent), it is required to announce the file name and/or download location; this simplifies the task of identifying the first seeder. However, announcements are not mandatory in protocols such as eDonkey, Gnutella and Gnutella 2. The publication of the availability of a file is performed via a searching mechanism within the protocols. Consequently, even if the keyword of a shared file is identified in a forum, no direct link can be drawn between the forum and the first seeder as in BitTorrent.

The identification of the first seeder is also affected by the time when the search function is initiated. If a single seeder is found at the beginning of the file distribution process, the seeder is likely to be the first seeder. However, without a reference time for file publication, an investigator would not be able to confirm when the file distribution started and how long the seeder was active.

Recently, two methods for identifying an initial seeder (or one of the first few seeders) have been published. The first method (Method 1) [5] repeatedly issues identical query patterns submitted by requesting addresses within a short period of time in the Foxy network for the query hits of interest that are returned. The second method (Method 2) [6] is based on the assumption that the growth of seeders in a P2P network follows the cumulative amount of seeders ("seeder curve"). Method 2 engages two rules:

- **Rule 1:** If a seeding peer is found to be reachable and connectable during the slow-rising period, the seeding peer is the first uploader.
- **Rule 2:** If a seeding peer is found after the file distribution level-off period, it is impossible to confirm that the seeding peer is the first uploader. If the single seeder is found during the slow-rising period, then it is very likely that the single seeder is the first seeder.

2.2 Identification Challenges

Several experiments were performed to verify the accuracy of the two methods. The experiments were performed using a modified Shareaza client, an open source P2P client that supports the Gnutella 2 protocol [8]. During the Foxy file sharing process, a SHA-1 hash value generated by the client is used as the identity of the file being shared. Thus, when the SHA-1 hash value is found, the associated file has already been uploaded.

As reported elsewhere [5, 6], our experiments confirmed that the number of query packets increase shortly before and after the appearance of the first seeder. Increases in the number of queries with SHA-1 values generated from multiple IP addresses were also observed. This proves that Method 1 could be used to identify the first seeder. However, there are two practical difficulties in implementing this method:

- Numerous hash values are generated in the Foxy network each day. We collected 100,000 to 650,000 new hash values per day from our monitored hubs alone. Monitoring all the hash values in the network would require a massive amount of computational resources.
- The growth rate of duplicate hash values is low. The number of identical hash values recorded for some files is as low as three or four hits over a 24-hour period, unless the files are very popular. Continuously monitoring all the files in the Foxy network would require significant resources.

It would appear that Method 2 is practical because the investigator only has to determine if the seeder is found in the slow-rising period. However, several questions must be answered:

- How can one confirm that the network monitoring was performed before the file of interest was widely distributed?
- How likely is the identified seeder one of the first few seeders?
- How can one confirm that the seeder was, in fact, identified during the slow-rising period?

For these reasons, Methods 1 and 2 may not be completely applicable to the Foxy network.

2.3 Simulation Challenges

Observations of file distribution in the Foxy network for clients using Gnutella 2 are not easily performed. The Gnutella 2 protocol used in Foxy is a decentralized P2P protocol. Every Foxy hub behaves as a searching server, so the returned results may only represent a portion of the search results from the entire Foxy network. Even if a suspected seeder is spotted, it is only possible to observe the localized query hit results. Also, it is not possible to confirm if the suspected first seeder is truly the first seeder or just one seeder in the swarm of seeders in the entire Foxy network during the file sharing process.

Peer nodes enter and leave the Foxy network very frequently. However, when a hub node leaves Foxy, the connected nodes restructure themselves by connecting to hubs and leaf nodes. This restructuring affects the search results returned to a leaf node.

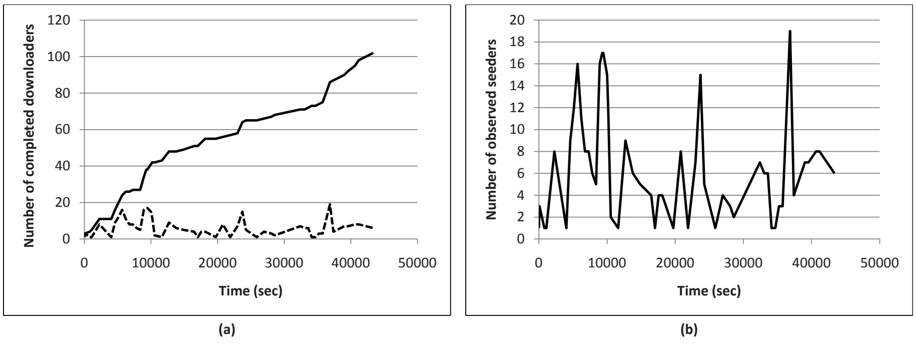


Figure 1. Seeder curves observed for a Shareaza-Foxy client.

A potential problem arises when a file has already been broadcasted and downloaded in an unmonitored part of the Foxy network. An identified sole seeder from a Foxy client could be one of the completed downloaders in the unmonitored network. If it recently connected to a monitored client as a seeder, it should not be considered as the first seeder.

Figure 1 presents the seeder curves for a Shareaza-Foxy client. Figure 1(a) shows the cumulative number of seeders (seeder curve) for a small but popular file (top plot). Note that the cumulative number of seeders increases throughout the period. The bottom plot (dashed line) in Figure 1(a) shows the actual number of seeders for the same file; its "zoomed-in" version is shown in Figure 1(b). The plot in Figure 1(b) shows nine instances where the number of observed seeders is one. However, if the number of seeders really drops to one, the overall download rate experienced by downloaders should be much less than the observed rate.

No matter how extensive the experiment, the number of hubs that can be monitored is only a small fraction of the total number of hubs in the Foxy network. Consequently, we decided to observe the file sharing behavior in a simulated Foxy network. By controlling the simulation environment, the behavior during the slow-rising period can be analyzed.

3. Simulation Experiments

This section describes the simulation experiments involving the Foxy network.

3.1 SimFoxy

Several researchers have conducted simulations of P2P networks [4, 7]. However, most of them use queuing models to simulate P2P network performance or focus on the Gnutella and BitTorrent protocols rather than the Gnutella 2 protocol.

Instead of relying on existing simulation programs, we built our own program, SimFoxy, to simulate the behavior of Foxy network clients. SimFoxy focuses on the download process after a file has been identified. It may be used to simulate a Foxy network that shares a single file from a single seeder with a predefined number of peers. The numbers of seeders and peers are recorded throughout a simulation. Various parameters (e.g., file packet number, packet size, upload and download connection limits, upload and download rates, etc.) that affect file propagation behavior are implemented as adjustable parameters in SimFoxy.

3.2 Simulation System

In most P2P downloads, file sharing initiation can be modeled as discrete events with Poisson inter-arrival times [4, 7]. The simplest way to analyze the initiation stage behavior of file distribution in the Foxy network is to develop a Foxy simulation environment using a discrete event simulation package. We employ SimFoxy, a Python-based implementation that uses the SimPy simulation module [9]. SimPy is an object-oriented, process-based discrete event simulation language developed in Python 2.x that supports the simulation of multiple processes.

All file sharing and download processes in SimFoxy are simulated at the packet level. A packet object is the basic unit that is uploaded and downloaded during the file sharing process. The packet-level simulation is performed by limiting the resource capacity of packets such as download connections by adding the amount of time expected to be used in the download process. Download activities are initiated by packets after receiving requests from peers. After one download activity involving a packet is completed, the packet pauses until it is invoked by another downloading peer.

The downloading and uploading of packets by a peer are regulated by the server object. To simplify the architecture and to reduce resource usage, instead of simulating the server as a hub in the Foxy network, the entire peer list, partial peer list, seeder list and peer availability for downloading are added as accessible resources in the server object. This reduces the resources required for simulation.

3.3 Simulation Assumptions

Our SimFoxy implementation incorporated several assumptions to ensure that it would be possible to analyze the effects of various environmental parameters in the Foxy network. First, we assumed that one original seeder exists in the simulated Foxy network and no new seeder connects to the Foxy network during the simulation. Second, all the peer nodes participate in uploading and downloading the target file only; this reduces the effect on the download bandwidth due to the distribution of other files. Third, all peers are only able to upload the file as a seeder after they have completely downloaded all the packets from the seeder; this captures the behavior of a Foxy 1.9.7 client, for which partial downloads of incomplete seeds are not supported.

The purpose of the simulation was to study the initial stage of peer upload during file distribution. In order to shorten the simulation time, we concentrated on the simulation of the first 100 to 1,000 peers during the upload and download periods. Consequently, in the simulations, the maximum number of preset peers in SimFoxy was limited to 1,000.

The upload connectivity and download connectivity are preset in Foxy clients. To reduce the complexity of downloads, all the Foxy clients should be configured to support a maximum of five downloads and ten uploads. These values were tested in our simulation experiments.

The downloading of file fragments in the network is controlled by the Foxy client. To simplify the simulation setup, download and upload fragments were defined to be 500 KB per packet, which is the packet size observed in the real Foxy download packet request query. Therefore, the complete download of a 10 MB file requires twenty 500 KB packets to be downloaded by a node.

The connection speed between an uploader node A and a downloader node B is assumed to be the minimum of the upload rate of A and the download rate of B. Usually, this is the upload speed of node A because the upload speed is normally less than the download speed. When one additional node is connected to node A, the upload speed is divided equally among the two nodes.

3.4 Simulation Sets

More than 100 simulation experiments were performed using SimFoxy. The simulations were performed by varying five parameters: (i) average inter-arrival time (T_{arr}); (ii) number of peers interested in the target file during the simulation period (N_p); (ii) simultaneous upload peer limit (N_u); (iii) simultaneous download peer limit (N_d); (iv) average inter-departure time (T_{dep}); and (v) file size expressed as the number of 500 KB packets (N_{pkt}). The upload and download rates of all the peers and the seeder were set to 1,280 KB/s (1 Mbps) and 2,560 KB/s (2 Mbps), respectively. Fixing the upload and download rates of all the peers reduces the effect of randomness on the parameter measurements.

The simulation experiments were divided into four sets defined below.

Set 1 This set of simulations investigated the effects of changes in the file size (i.e., number of packets (N_{pkt})). In Sets 1(a), 1(b) and 1(c), experiments were performed by varying N_{pkt} only. In Set 1(d), N_{pkt} was fixed, but different upload (N_u) and download (N_d) limits were used. The inter-arrival (T_{arr}) and inter-departure (T_{dep}) times were set to 5 seconds and 10 seconds, respectively.

- **Set 1(a):** Simulations involving different numbers of peers; N_{pkt} = 20 (~10 MB); N_p = 1 to 1,000; N_u = 5; N_d = 10.
- **Set 1(b):** Simulations involving different numbers of packets; N_{pkt} = 20 to 400 (~200 MB); N_p = 5; N_u = 10.
- **Set 1(c):** Simulations involving the sharing of large files; N_{pkt} = 800 (~0.4 GB), 1,600 (~0.8 GB), 3,200 (~1.6 GB); N_p = 1, 3, 4; N_u = 5; N_d = 10.
- **Set 1(d):** Simulations involving different upload and download limits; N_{pkt} = 20; N_p = 1, 2, 5, 10, 25, 50; N_u = 5, 10, 20, 40; N_d = 10, 20, 40, 50.

Set 2 This set of simulations investigated the effects of changes in the inter-departure time (T_{dep}) after download completion. In Sets 2(a), 2(b), 2(c), 2(d) and 2(e), experiments were performed by varying T_{dep} from 10 to 1,200 seconds with the upload (N_u) and download (N_d) limits fixed at 5 and 10, respectively; the number of packets (N_{pkt}) limited to 100 pieces; and the number of peers (N_p) fixed at 100 nodes.

- **Set 2(a):** Simulation involving different inter-departure times; N_{pkt} = 100; N_p = 100; N_u = 5; N_d = 10; T_{dep} = 10 seconds.
- **Set 2(b):** Simulation involving different inter-departure times; N_{pkt} = 100; N_p = 100; N_u = 5; N_d = 10; T_{dep} = 400 seconds.
- **Set 2(c):** Simulation involving different inter-departure times; N_{pkt} = 100; N_p = 100; N_u = 5; N_d = 10; T_{dep} = 800 seconds.
- **Set 2(d):** Simulation involving different inter-departure times; N_{pkt} = 100; N_p = 100; N_u = 5; N_d = 10; T_{dep} = 1,000 seconds.
- **Set 2(e):** Simulation involving different inter-departure times; N_{pkt} = 100; N_p = 100; N_u = 5; N_d = 10; T_{dep} = 1,200 seconds.

Set 3 This set of simulations investigated the effects of changes in the inter-arrival time (T_{arr}). In Sets 3(a), 3(b), 3(c) and 3(d), experiments were performed by varying the inter-arrival time patterns (all at once, periodic, random and uniform random) and sharing 100 packets with 100 peers with an inter-departure time (T_{dep}) of 1,000 seconds. The upload (N_u) and download (N_d) limits were fixed at 5 and 10, respectively.

- **Set 3(a):** Simulation involving 100 peers downloading simultaneously; $T_{arr} = 0$ seconds.
- **Set 3(b):** Simulation involving 100 peers starting their downloading at different inter-arrival times; $T_{arr} = 5, 10, 20, 40$ seconds.
- **Set 3(c):** Simulation involving 100 peers starting their downloading at random times; T_{arr} = random: 0 to 1,200 seconds.
- **Set 3(d):** Simulation involving 100 peers starting their downloading at uniform random times; T_{arr} = uniform random: 0 to 1,200 seconds.

Set 4 This set of simulations investigated the effects of changes in the inter-arrival time (T_{arr}) patterns (periodic, random, uniform random and Poisson random). In Sets 4(a), 4(b) and 4(c), experiments were performed by sharing 200 pieces of packets (N_{pkt}) with different inter-arrival time patterns (all at once, random and uniform random). In Sets 4(d) and 4(e), experiments were conducted by sharing of 20 pieces of packets with $T_{dep} = 1$ to 10 seconds and $T_{arr} = 100$ seconds; and by sharing 20 and 40 pieces of packets with Poisson random T_{arr} ($\lambda = 0.25$), respectively. The upload (N_u) and download (N_d) limits were fixed at 5 and 10, respectively.

- **Set 4(a):** Simulation involving a popular file being downloaded by all the peers simultaneously; $N_{pkt} = 200$; $N_p = 100$; $T_{arr} = 0$ seconds; $T_{dep} = 100$ seconds.
- **Set 4(b):** Simulation involving random incoming peers; $N_{pkt} = 200$; $N_p = 100$; T_{arr} = random: 0 to 1,200 seconds; $T_{dep} = 100$ seconds.
- **Set 4(c):** Simulation involving uniform random incoming peers; $N_{pkt} = 200$; $N_p = 100$; T_{arr} = uniform random: 0 to 3,600 seconds; $T_{dep} = 0, 120, 2{,}000, 3{,}000, 8{,}000$ seconds.
- **Set 4(d):** Simulation involving slow inter-arrival times; $N_{pkt} = 20$; $N_p = 1$ peer/second, 1 peer/10 seconds; T_{arr} = periodic: 1 peer/second; $T_{dep} = 100$ seconds.

- **Set 4(e):** Simulation involving Poisson random incoming peers with different inter-departure times; $N_{pkt} = 20, 40$; $N_p = 100$; T_{arr} = Poisson random ($\lambda = 0.25$); T_{dep} = 100, 1,000 seconds.

These four sets of simulation experiments facilitated the systematic analysis of the effects of various parameters on the download duration, slow-rising period and the time required for the appearance of the second seeder.

3.5 Observations

Several observations can be made based on the simulation experiments.

Comparison of Experimental and Simulation Results Experiments performed in the actual Foxy network cannot reflect the file distribution over the entire network. Therefore, the behavior in the real and simulated Foxy networks may not match completely.

To demonstrate how closely SimFoxy simulates the real Foxy network, some simulations were conducted using parameters obtained from real-world environments. Actual Foxy network download scenarios were captured by conducting two file downloads at different instants. In both cases, observations based on our modified Foxy client revealed that the incident was initiated by one observable seeder. Following this, seeder growth curves were constructed using SimFoxy with similar criteria.

Figure 2 shows four seeder curves for real and simulated Foxy networks. Figure 2(a) shows the observed number of seeders (dashed line) and the cumulative number of seeders (unbroken line) for a small, popular file. Figure 2(b) shows the simulation results for a small to medium sized file with a rapid arrival rate in SimFoxy. Figure 2(c) shows the behavior during the first 12 hours for a large, popular file in the Foxy network. Figure 2(d) shows the simulation results for a large file with slow peer departure and rapid arrival rates in SimFoxy. Note that the curves in Figures 2(b) and 2(d) match the majority of the actual completed downloader growth rate curves (dashed lines) in Figures 2(a) and 2(c).

Relationship between File Size and Download Time After clarifying the relationship between file size and the download completion time of the first downloader, we attempted to determine how the number of competing peers affects the download time based on the results obtained in Sets 1(c) and 1(d).

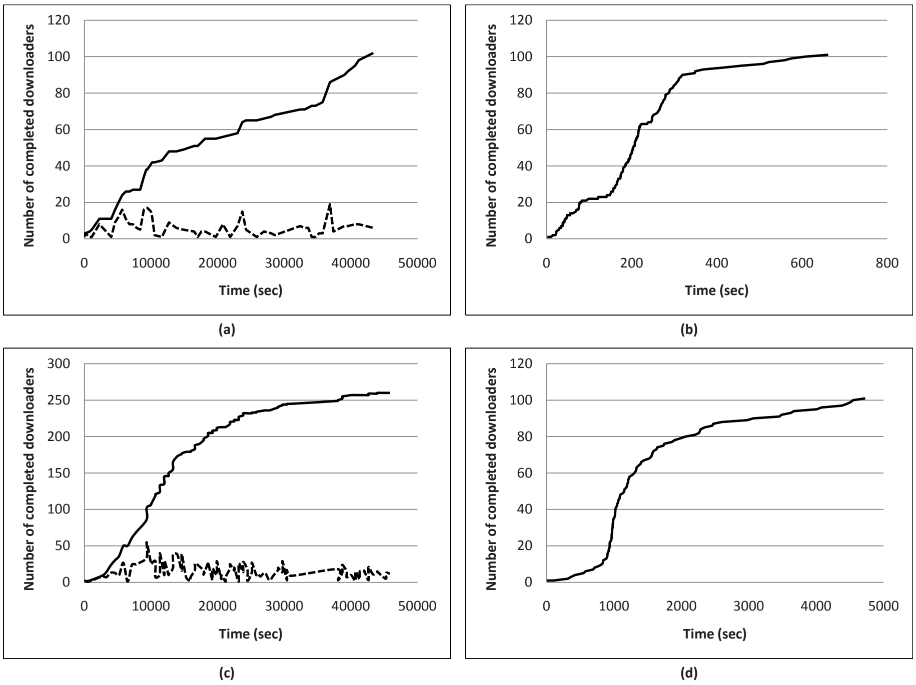
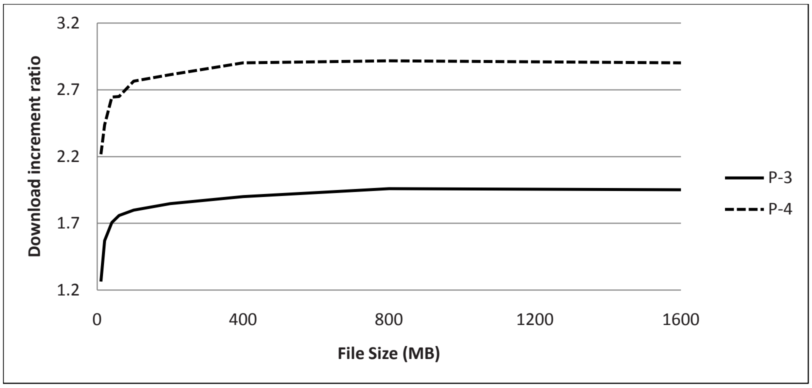


Figure 2. Four seeder curves for real and simulated Foxy networks.

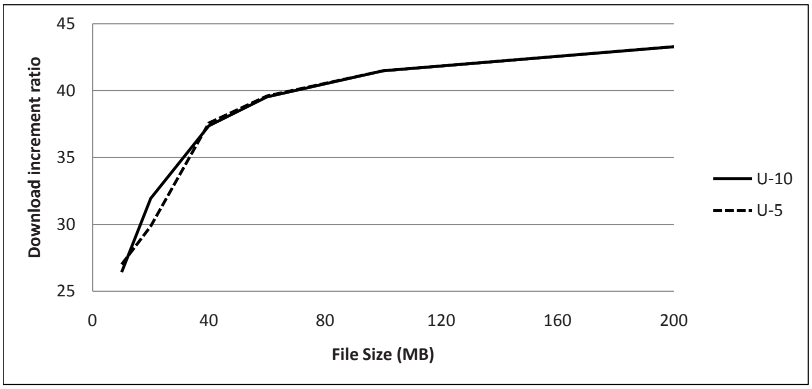
Th download time is affected by the number of peers competing for a single file as well as by the file size (Set 1 simulations). If the file download duration for one downloader is T_x, the increment due to additional peers, which we call the "download increment ratio" (R) is $(T_y - T_x)/T_x$, where T_y is the download duration time when the number of downloaders is greater than one.

Instead of varying the file size, we plotted the download increment ratio against increments in file size for three peers and four peers downloading simultaneously (Figure 3(a)). The x-axis is R and the y-axis is the size of the target file (in MB). The curve in Figure 3(a) shows the effect of the number of peers competing for same file at one time (P_i means that i peers are competing). Note that R is affected by the file size but eventually levels off.

Effect of Upload Limit on Download Time Instead of comparing the number of competing peers, the download increment ratio R for different upload limits was compared based on the Set 1(d) results. The download increment ratio R corresponding to the same file source with the same competing peers was measured for two different upload limits



(a)



(b)

Figure 3. Download increment ratio versus file size.

(5 peers and 10 peers). Figure 3(b) highlights the effect of the upload limit on the download increment ratio (U_j means the upload limit is j peers). Note that the upload limit has almost no effect on the download increment ratio R.

Effect of Departure Rate on the First Downloader Completion Time The departure rate of peers affects the overall file distribution behavior. The Set 2 simulations show that the departure rate of peers definitely affects the download time of successive peers. With a higher departure rate, the download speed after the rapid-rising period would be greatly reduced. However, because the first downloader must complete the download from the first seeder, it is only affected by the behavior of the first uploader. Thus, the departure rate of peers was found to have no effect on the completion time of the first downloader.

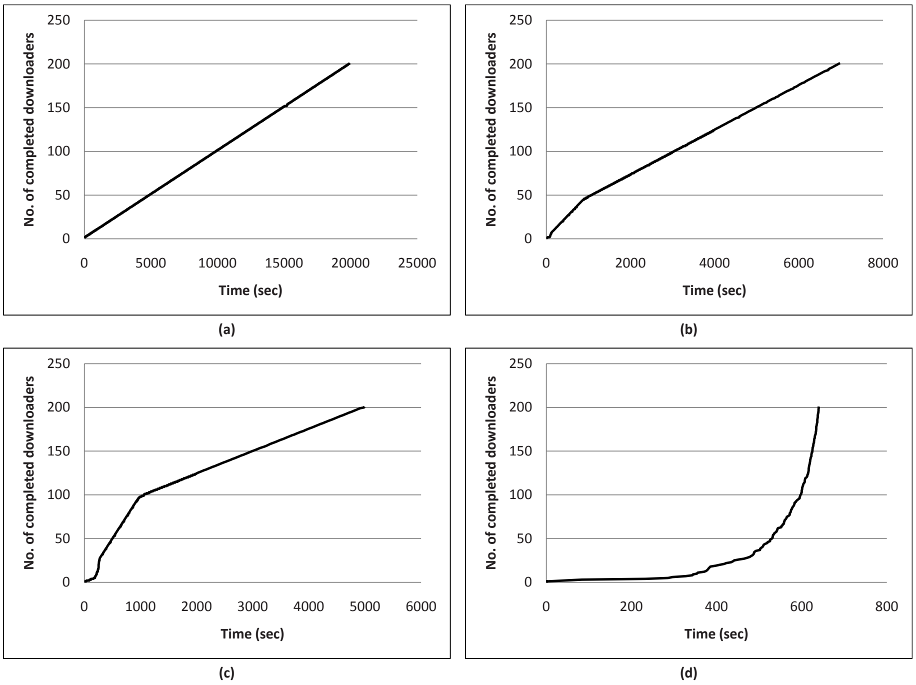


Figure 4. Cumulative number of downloaded copies over time.

Effect of Arrival Rate on Peer Download Time Using the data from the Set 3 and Set 4 simulations, the growth rate of seeders was measured against different peer inter-arrival times from 0 to 100 seconds. Figure 4 shows the cumulative number of downloaded copies over time for 200 peers initiated with different inter-arrival times (Figure 4(a): 100 seconds; Figure 4(b): 20 seconds; Figure 4(c): 10 seconds; Figure 4(d): 0 seconds). The download duration is not affected by the peer inter-arrival time if the inter-arrival time is greater than the download duration. However, when the inter-arrival time is reduced, the download duration is affected and the relationship changes from a straight line to a curve (Figure 4).

Effect of Peer Download Time Variation Instead of simply measuring the growth rate of seeders as in the Set 3 simulations, we measured the first connected time of peers and calculated their file download completion duration (download duration). In the Set 4 simulations, all the peers had the same simulated upload and download speeds and the download duration was plotted against the first connected time as in Figures 5 and 6. The two figures show the cumulative number of seeders

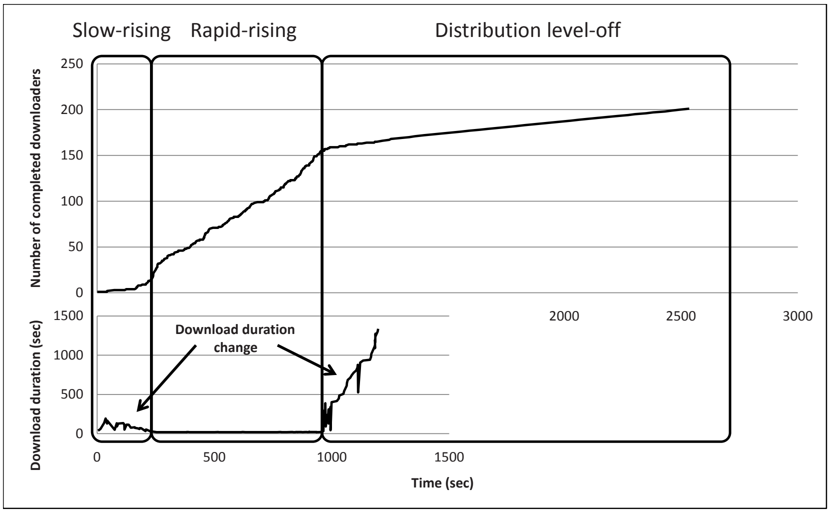


Figure 5. Downloader (upper) and download duration curves (lower) (random).

(seeder curve) and the corresponding download duration curve for two peer inter-arrival patterns. Figure 5 shows the peer download behavior when the peers arrive randomly between 0 to 1,200 seconds based on Set 4(b).

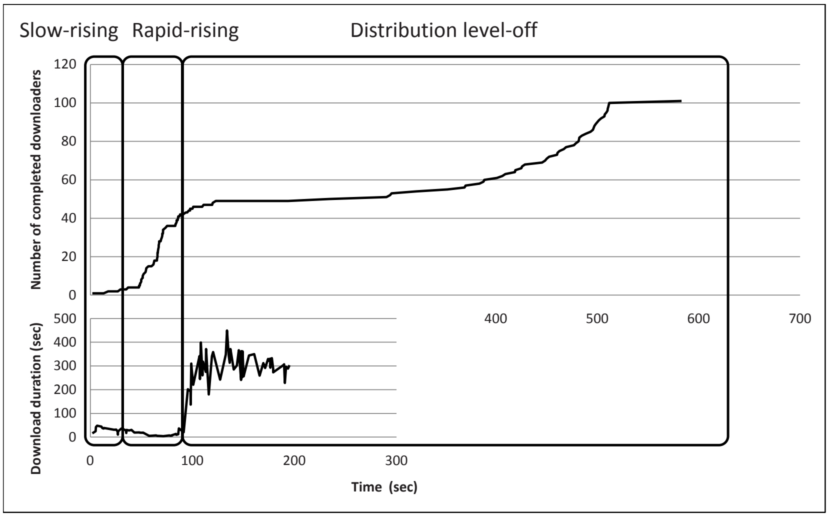


Figure 6. Downloader (upper) and download duration curves (lower) (Poisson).

Figure 6 shows the peer download behavior for a Poisson inter-arrival time [3] with $\lambda = 0.25$ according to Set 4(e). The value of λ was set to

0.25 to ensure that the peers connected to the network for download were spread across the entire download period instead of being concentrated at the beginning of the download period. Note that the download duration of peers exhibits a strong relationship with the slow-rising period and the rapid-rising period.

The download duration (lower) curves in Figures 5 and 6 show that the download time reduces throughout the slow-rising period. The download duration is rather short throughout the rapid-rising period, after which it rises when peers that complete their downloads leave the Foxy network.

4. Simulation Results

This section discusses the main results obtained in the simulation experiments.

The download duration change is another indicator of the slow-rising period. Figures 5 and 6 show that the download duration T_d is affected by the time when the download request is initiated. During the slow-rising period, peers experience long download times and many download interruptions because a single file resource is shared by multiple peers $P_1, \ldots, P_n$.

When multiple peers provide a resource, a peer P_i may obtain packets from peers other than the seeder S and the download duration T_d is reduced. At the same time, the number of completed downloaders increases much faster – this is observed as the rapid-rising period. The download duration remains approximately the same because the download connectivity is pre-defined in the client.

The equilibrium is disrupted when more peers disconnect after completing their downloads than the increment in the number of completed downloaders. As the number of available seeders goes down, T_d increases again until no more peers join or leave the Foxy network.

The popularity of a resource increases the first peer download time. Analysis of the simulation results reveals that the peer inter-arrival time and the file size greatly affect the seeder growth rate. When no peers compete for the same seeder, the peer download time is essentially the same as the time required to download the resource from a single server. As more peers download from a source simultaneously, the download completion time increases.

In the actual Foxy network, peers search for a popular file after it is published and announced. Peers $P_1, \ldots, P_n$ could attempt to connect to the same file source simultaneously. Seeder S uploads a packet PK_j to Peer P_i when the request for PK_j by P_i is accepted by S.

However, the acceptance of a request is limited by the upload limit of S. When the number of requests from peers P_i, ..., P_n exceeds the maximum upload limit, the peers whose requests were accepted earlier by S are granted packet PK_j while the other peers have to wait until the earlier download requests are completed. As more peers with same download rate request the same file from S, the probability of obtaining PK_j by P_i drops. Thus, the download duration of the second seeder from the first seeder is increased. According to our experiments, if all the peers who request the file have the same configuration, then the number of peers is directly proportional to the lengthening of the download duration.

5. Seeder Identification Rules

A single seeder in a Foxy network can be identified, but verifying that the seeder is one of the initial seeders is not a simple task. In our previous research [6], we showed that a seeder can be identified as one of the first few seeders if the sharing of the resource falls within the slow-rising period of the seeder curve. However, confirming whether or not the period of interest falls within the slow-rising period is also difficult.

Our simulation results reveal that the download duration reflects the behavior of seeder curve. Instead of observing the number of seeders in the Foxy network, investigators could perform multiple downloads of the same file at different times from different clients. Then, the results could be analyzed using the following rules.

- **Rule 1:** If two or more observed download durations T_d drop during consecutive downloads, the observed seeders should be collected within the slow-rising period.

- **Rule 2:** If T_d remains roughly steady at the stable download duration, the observed seeders should be collected during the rapid-rising period.

- **Rule 3:** In the case of a popular file, the download duration T_d for the second seeder is lengthened. This duration is directly proportional to the number of peers that simultaneously download the file. The slow-rising period is lengthened by the number of requesting peers. Therefore, the period for identifying initial seeders is lengthened by the popularity of the file.

- **Rule 4:** If the file download completion time T_d is less than the peer inter-arrival time T_{arr}, then it is impossible to confirm the appearance of first seeder.

Rules 1 and 2 can help confirm that the seeder is collected during the slow-rising period. Rules 3 and 4 can help verify that the chance of mistakenly identifying the seeder as the first seeder is reduced.

6. Conclusions

The identification of the first seeder is a crucial task in investigations of illegal file sharing in P2P networks. The approach for identifying first seeders in the popular Foxy network based on the slow-rising period of the cumulative seeder curve can be very helpful in investigations. Furthermore, rules derived from the relationships between key parameters – such as file popularity, number of packets and the maximum upload limit when the first seeder is connected to the network – assist investigators in determining if an identified seeder is, in fact, the first seeder.

The work presented in this paper is experimental in nature. Our future research will develop and validate a mathematical model that expresses the relationships between the number of hubs, number of peers, seeder growth rate and download duration. Such a model would support network forensic investigations as well as the design and implementation of strategies for controlling illegal file sharing in P2P networks.

References

[1] K. Chow, K. Cheng, L. Man, P. Lai, L. Hui, C. Chong, K. Pun, W. Tsang, H. Chan and S. Yiu, BTM – An automated rule-based BT monitoring system for piracy detection, *Proceedings of the Second International Conference on Internet Monitoring and Protection*, p. 2, 2007.

[2] K. Chow, R. Ieong, M. Kwan, P. Lai, F. Law, H. Tse and K. Tse, Security Analysis of the Foxy Peer-to-Peer File Sharing Tool, Technical Report TR-2008-09, Department of Computer Science, Hong Kong University, Hong Kong, 2008.

[3] P. Consul, *Generalized Poisson Distributions: Properties and Applications*, Marcel Dekker, New York, 1989.

[4] B. Fan, D. Chiu and J. Lui, Stochastic differential equation approach to model BitTorrent-like P2P systems, *Proceedings of the IEEE International Conference on Communications*, pp. 915–920, 2006.

[5] R. Ieong, P. Lai, K. Chow, F. Law, M. Kwan and K. Tse, A model for Foxy peer-to-peer network investigations, in *Advances in Digital Forensics V*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 175–186, 2009.

[6] R. Ieong, P. Lai, K. Chow, M. Kwan, F. Law, H. Tse and K. Tse, Forensic investigation and analysis of peer-to-peer networks, to appear in *Handbook of Research on Computational Forensics, Digital Crime and Investigation: Methods and Solutions*, C. Li (Ed.), Information Science Reference, Hershey, Pennsylvania, 2010.

[7] D. Qiu and W. Sang, Global stability of peer-to-peer file sharing systems, *Computer Communications*, vol. 31(2), pp. 212–219, 2008.

[8] Shareaza, Shareaza 2.5.2.0 (shareaza.sourceforge.net), 2010.

[9] SimPy, SimPy Simulation Package (version 2.0.1) (simpy.sourceforge.net), 2009.

IV

LIVE FORENSICS

Chapter 12

UNCERTAINTY IN LIVE FORENSICS

Antonio Savoldi, Paolo Gubian and Isao Echizen

Abstract The goal of live digital forensics is to collect crucial evidence that cannot be acquired under the well-known paradigm of post-mortem analysis. Volatile information in computer memory is ephemeral by definition and can be altered as a consequence of the live forensic approach. Every running tool on an investigated system leaves artifacts and changes the system state. This paper focuses on the understanding and measurement of the uncertainty related to the important and emerging paradigm of live forensic investigations. It also presents some practical examples related to the evaluation of uncertainty.

Keywords: Live forensics, collection tools, measurement of uncertainty

1. Introduction

When performing live forensics, a practitioner is expected to run tools, toolkits and/or custom live distributions on a powered computer system. This is especially the case when a timely triage or mission-critical analysis is required [19]. Indeed, apart from the well-known post-mortem paradigm, which has been used for many years, live forensics is emerging as the new standard for dealing with large, mission-critical computer systems [19].

The goal of live forensics is to collect evidentiary data from computer memory in order to define the state of the computer system at the time of an incident. Naturally, the act of collecting data from a live system causes changes to the volatile memory [10]; consequently, it is important that the forensic practitioner documents and explains the possible artifacts in the resulting digital evidence. For example, running a popular tool such as `dd` [5] from a removable media device alters volatile data as soon as the program is loaded in main memory. Another example is the Helix tool [4], which may create or modify files and registry entries on the system being investigated. Similarly, a remote forensic tool necessarily establishes a network connection, executes instructions in memory and makes other alterations to the system.

From a forensic standpoint, the acquisition process should modify the original system memory state as little as possible, and all changes should be documented and assessed in the context of the final analytical results. However, it is controversial to measure how much of the volatile memory is modified by a collection tool whose goal is to acquire the full volatile memory content. Moreover, it is difficult (if not impossible) to establish the extent of the "perturbation" caused by a running process on volatile memory. As a consequence, it is necessary to design a sound and reliable methodology that measures the volatile memory changes caused by a forensic tool or toolkit running on a live system.

This paper describes a methodology for measuring the uncertainty in live forensics. The application of the methodology is illustrated using collection tools targeted for live Windows and Linux systems.

2. Background

Uncertainty is a term that is used in subtly different ways in fields such as philosophy, physics, statistics, economics, finance, insurance, psychology, sociology, engineering and information science. It applies to predictions of future events, to physical measurements already made, or to the unknown. Although the term is used in a variety of ways in public discourse, specialists in decision theory, statistics and other quantitative fields have defined uncertainty and risk more specifically. Hubbard [6, 9] defines uncertainty and the related concept of risk as follows:

- **Uncertainty:** Uncertainty is the lack of certainty. It is a state of having limited knowledge where it is impossible to exactly describe the existing state or future outcome or more than one possible outcome.

- **Measurement of Uncertainty:** The uncertainty of a set of possible states or outcomes is measured by assigning probabilities to each possible state or outcome; this also includes the use of a probability density function for continuous variables.

- **Risk:** Risk is a state of uncertainty where some possible outcomes have an undesired effect or significant loss.

- **Measurement of Risk:** Risk is measured in terms of a set of uncertainties where some possible outcomes are losses and in terms of the magnitudes of the losses; this also includes the use of loss functions for continuous variables.

From an engineering perspective, uncertainty is characterized as Type A or Type B, depending on the method used to evaluate it. A Type A method evaluates uncertainty using a statistical analysis of a series of observations. On the other hand, a Type B method evaluates uncertainty by a means other than the statistical analysis of a series of observations. These two components of the uncertainty can be quantified by the statistically estimated standard deviation, u_i and u_j, for Type A and Type B methods, respectively [14].

To illustrate Type A uncertainty, consider an input quantity X_i whose value is estimated from n independent observations $X_{i,k}$ of X_i obtained under the same measurement conditions. In this case, the input estimate x_i is usually the sample mean given by:

$$x_i = \overline{X_i} = \frac{1}{n}\sum_{k=1}^{n} X_{i,k}$$

The standard uncertainty $u(x_i)$ associated with x_i is the estimated standard deviation of the mean:

$$u_{x_i} = \left(\frac{1}{n(n-1)}\sum_{k=1}^{n}(X_{i,k} - \overline{X_i})^2\right)^{1/2}$$

A Type B evaluation of standard uncertainty is usually based on a scientific judgment using all the relevant information that is available. This may include: (i) previous measurement data; (ii) experience with, or general knowledge of, the behavior and property of relevant materials and instruments; (iii) manufacturer specifications; (iv) data provided in calibration and other reports; and (v) uncertainties assigned to reference data taken from handbooks.

Type B uncertainty is often quite difficult to measure. As a consequence, it is useful to establish an upper-bound measurement of the uncertainty when performing a live forensic analysis that involves the execution of one or more tools on a target system that affects the live memory.

3. Measuring Uncertainty in Digital Forensics

Casey [3] proposed that the uncertainty for network-related evidence be calculated by estimating probability distributions. Given a series of real measurements (e.g., clock offsets from a network of computers), it is possible to estimate the real distribution of data. If a normal or Gaussian data distribution is observed, an accuracy measurement for the data set under investigation can be obtained. The precision of the measurement can be described using the mean μ and standard deviation σ. Although, this statistical method is a relatively straightforward way to estimate the uncertainty, it is not necessarily applicable or accurate and it strongly depends on the kind of data that is available.

In order to measure how much a running forensic tool affects system memory, it is necessary to define a methodology that quantifies how much the volatile memory changes. This is closely related to the concept of a "footprint," which measures the amount of volatile memory that a program uses or references when running. The footprint includes all active memory regions such as code, static data sections (initialized and uninitialized), and heap and stack areas. In addition, it includes the memory required to hold data structures such as symbol tables, constant tables, debugging structures and open files that the program requires while executing and that are loaded at least once during its execution [15]. In the case of Microsoft Windows, it is not possible to determine the footprint by examining the working memory (heap memory) using the Windows Task Manager (WTM). For example, WTM is unable to properly quantify all the kernel space memory that is affected by a tool like `dd`. In addition, effects such as the paging of less used memory pages to disk may occur when running the collection tool.

A promising approach is to sample the volatile memory (RAM contents) before and after a forensic tool runs and to repeat the procedure several times to account for statistical variations of the measurements. The collection tools used would depend on the specific operating system running on the live machine. Tools for Windows systems include `dd` [5], `mdd` [12], FastDump (`fd`) [7], `win32dd` [20] and Memoryze [11]. Tools for Linux platforms include GNU `dd` [16] and `dc3dd` [17]. These collection tools also probe the running live system. Ideally, such probes should not affect the volatile memory. Unfortunately, the available collection tools modify the volatile content to a greater or lesser extent.

The binary difference between memory snapshots (samples) collected by a tool (e.g., `dd`) at different times is a good estimator of the uncertainty introduced by the collection tool. Also, by taking a series of memory snapshots and measuring the statistical variation using the sample average (μ_{dmp}) and standard deviation (σ_{dmp}) of the different memory pages between consecutive memory snapshots, it is possible to define a more robust measurement of the uncertainty introduced by the tool. The uncertainty may be quantified as $U = \mu_{dmp} \pm \sigma_{dmp}$, where the sample average is related to the different memory pages between consecutive snapshots, and the sample standard deviation is related to the statistical error of the measurement. Figure 1 illustrates the timeline of an experiment required to evaluate the uncertainty of a collection tool.

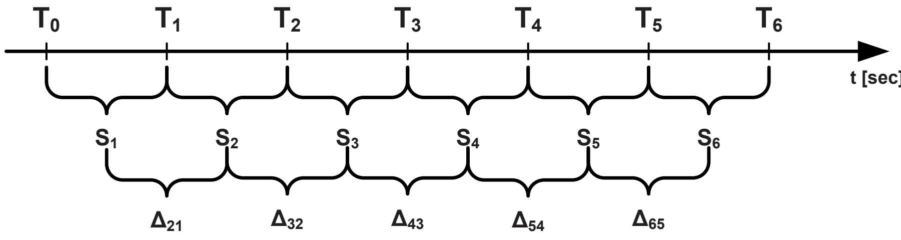


Figure 1. Timeline for evaluating the uncertainty of a tool.

The sampling procedure may be repeated as many times as possible in order to account for normal statistical variations.

Our definition of uncertainty requires the measurement of the real memory variations caused by a tool; in our case, the collection tool that is used as a probe to evaluate the uncertainty caused by other forensic tools. In this way, it is possible to show the extent of the "perturbation" caused by an imaging tool. Moreover, the practical effect of this variation can impact the integrity of the memory snapshot, which should not be altered according to digital forensic science principles.

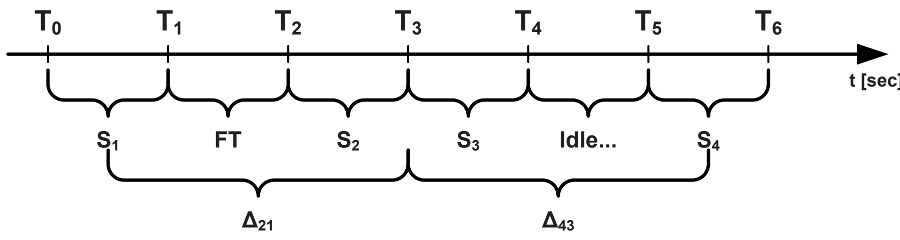


Figure 2. Timeline for evaluating the uncertainty of a live forensic tool.

This approach considers the ordinary uncertainty of the measurement tool (i.e., collection tool) as if it were the only tool running on the system. This assumption implies that the uncertainty of the probing tool should be determined only by the probing process (e.g., `dd`, `fd`, `mdd`, `win32dd` or Memoryze), without considering other processes (e.g., background processes) on the live system being examined. The resolution of the probing tool defines an upper bound on the precision of the measurement procedure. In fact, the probing tool itself affects the measurement procedure, and it is desired to have the best possible precision. On the other hand, in order to evaluate the uncertainty of a live forensic tool (e.g., Windows Forensic Toolchest [13]), it would be necessary to follow the procedure illustrated in Figure 2. The memory variation caused by the running toolkit is measured as the binary difference between the two snapshots S_1 and S_2, which is denoted as Δ_{21}. This represents an upper bound limit on the uncertainty. The lower bound is determined by the binary difference between S_3 and S_4 (say Δ_{43}), which takes into account the memory variations while the system is in an idle state (i.e., only background processes are running) for the period of time it takes to run the forensic toolkit. As a consequence, the uncertainty corresponding to the forensic toolkit is in the range $\Delta_{43} \leq U_{tot} \leq \Delta_{21}$. This approach may be used on live systems where it is not possible to acquire the memory by halting the system (as in virtual machines [1]). In fact, the ideal procedure would be to halt the system, take the memory snapshot using an atomic operation, run the forensic tool, and finally collect an additional memory snapshot after halting the system. At this time, such a procedure can be conducted only by using virtual technology.

4. Experimental Setup

Our experiments used several collection tools for two major operating systems, Windows XP SP3 and Linux Suse 11.1 (Kernel 2.6.27.7-9). The tools used on the Windows system were `dd`, `fd`, `mdd`, `win32dd` and Memoryze. The tools used on the Linux system were GNU `dd` [16] and `dc3dd` [17].

- **`dd` (version 1.0.0.1035)**
  - Usage: `dd.exe if=\\.\PhysicalMemory of=dump.bin conv=noerror`
- **`fd` (version 1.3.0)**
  - Usage: `fd.exe [-nodriver] dump.bin`
  - If no options are specified, the memory snapshot is collected by installing the `fd` driver. The `[-nodriver]` option causes the snapshot to be collected using a Windows API.
- **`mdd`**
  - Usage: `mdd.exe -o dump.bin`
- **`win32dd` (version 1.2.2.20090608)**
  - Usage: `win32dd.exe -l 0 -r dump.bin`
  - The `-r` option produces a raw binary image of the memory content. The `-l 0` option accesses the `\\Device\\Physical Memory` device. The `-l 1` option uses `Windows Kernel API MmMapIoSpace()`.

- **Memoryze**
  - Usage: `MemoryDD.bat`
  - The `-offset` option specifies the offset into physical memory. The `-size` option specifies the size of physical memory to acquire. The `-output` specifies the directory where the results are written (default "Audits").
- **GNU** `dd`
  - Usage: `dd if=/dev/mem of=/mnt/sdb1/dump.bin conv=noerror`
- **dc3dd**
  - Usage: `dc3dd if=/dev/mem of=/mnt/sdb1/dump.bin conv=noerror`

5. Experimental Results

This section describes the results of evaluating the uncertainty of collection tools in Windows XP SP3 and Linux (OpenSuse 11.1) systems. As described in Section 2 and in our previous work [18], the evaluations involve the analysis of the real footprints of running tools.

5.1 Windows System

This section presents the results obtained for five collection tools that were run on a Dell Optiplex 330 platform (1 GB RAM) under Windows XP SP3. Table 1 presents the uncertainty for the five collection tools; note that `fd` was run under two operational modes `fd1` and `fd2`. Ten memory snapshots (S = 10) of size MP were collected using each tool/ operational mode (MP denotes the number of collected 4 KB memory pages). Each snapshot was copied to an external USB 2.0 drive. μ_{dmp} denotes the sample mean of the different (4 KB) memory pages and σ_{dmp} denotes the sample standard deviation for consecutive memory snapshots. The number of 4 KB memory pages, size in MB and percentage are provided for μ_{dmp} and σ_{dmp}. T_{coll} lists the average collection times and U_{tot} lists the uncertainty measurement, which is expressed as a percentage of the modified volatile memory. For example, $U = 65.5\% \pm 0.15\%$ quantifies the uncertainty of the memory snapshot collected with the `dd` tool. The result is quite surprising in that more than 65% of the total memory is affected during the collection process.

Table 1 also shows that while the uncertainty for `win32dd` ($U_{\texttt{win32dd}} = 9.6\% \pm 0.42\%$) is the lowest among the tested tools, the corresponding

Table 1. Tool uncertainty for a Windows system.

Tool	**S**	**MP**	μ_{dmp}			σ_{dmp}			T_{coll}	U_{tot}
	[n°]	**[n°]**	**[n°]**	**[MB]**	**[%]**	**[n°]**	**[MB]**	**[%]**	**[sec]**	**[%]**
`dd`	10	259,417	169,835	663.41	65.5	389	1.52	0.15	55	65.5 ± 0.15
`fd1`	10	259,287	168,811	659.42	65.1	346	1.35	0.13	55	65.1 ± 0.13
`fd2`	10	259,287	167,865	655.72	64.7	1181	4.61	0.45	56	64.7 ± 0.45
`mdd`	10	259,287	167,945	656.03	64.8	360	1.41	0.14	42	64.8 ± 0.14
`win32dd`	10	259,428	249,51	97.46	9.6	1093	4.27	0.42	447	9.6 ± 0.42
Memoryze	10	259,417	161,672	631.53	62.3	406	1.59	0.16	60	62.3 ± 0.16

time (T_{coll}) is approximately eight times higher. This is not the ideal case because the memory snapshot should be collected as an atomic operation, as fast as possible and with minimum integrity leaks. Indeed, the results show that a trade-off exists between uncertainty and speed.

Table 2. Tool uncertainty for a virtual Windows system.

Tool	**S**	**MP**	μ_{dmp}			σ_{dmp}			T_{coll}	U_{tot}
	[n°]	**[n°]**	**[n°]**	**[MB]**	**[%]**	**[n°]**	**[MB]**	**[%]**	**[sec]**	**[%]**
GNU `dd`	10	262,143	189,762	741.25	72.4	4,689	18.32	1.79	93	72.4 ± 1.79
`fd1`	10	262,010	187,930	734.10	71.7	270	1.05	0.10	90	71.7 ± 0.10
`fd2`	10	262,010	186,482	728.44	71.2	958	3.74	0.36	92	71.2 ± 0.36
`mdd`	10	262,010	238,482	931.57	90.1	100	0.39	0.038	74	90.1 ± 0.038
`win32dd`	10	262,144	22,589	88.24	8.6	714	2.79	0.27	2,059	8.6 ± 0.27
Memoryze	10	262,143	199,832	780.64	76.2	146	0.57	0.056	99	76.2 ± 0.056

Table 2 presents the results obtained for a virtual Windows XP SP3 based system (1 GB RAM). Note that the virtual system was customized in the same manner as the real Windows system (Table 1) in order to provide an almost identical testing environment. The snapshots were collected using the same tools and copied to an external USB 2.0 hard disk. The uncertainty values shown in Table 2 are on the average 10% higher for the `dd`, `fd1`, `fd2` and Memoryze tools compared with the results in Table 1. Also, the uncertainty for the `mdd` tool is 30% higher and the collection time has increased by 30%. Interestingly, the uncertainty for `win32dd` has remained almost constant, whereas its collection time has increased about 8.5 times to 2,059 seconds.

Clearly, an ideal probing tool should have the fastest collection time but should affect the memory snapshot as little as possible. However, the results show that this may not be appropriate in the case of a virtual system, where it would be more precise to sample the memory from outside the system.

Table 3. Tool uncertainty for a virtual Linux system (no direct I/O).

Tool	S	MP	μ_{dmp}			σ_{dmp}			T_{coll}	U_{tot}
	[n°]	[n°]	[n°]	[MB]	[%]	[n°]	[MB]	[%]	[sec]	[%]
GNU `dd`	10	131,072	32,032	125.25	24.4	969	3.78	0.74	25	24.4 ± 0.74
`dc3dd`	10	131,072	33,134	129.43	25.3	1123	4.38	0.10	28	25.3 ± 0.10

5.2 Linux System

This section presents the experimental results obtained for two collection tools that were run on a virtual Linux system (Suse Linux 11.1; Kernel 2.6.27.7-9) equipped with 512 MB RAM and a 10 GB hard disk. The two tools used were GNU `dd` and `dc3dd`. An interesting feature of the Linux OS is that it bypasses the memory cache system [2]. This makes it possible to transfer a memory snapshot with direct I/O communication, resulting in better memory coherence of the snapshot (i.e., lower uncertainty than when the page cache system is used). This feature is not available for the Windows OS.

Table 3 summarizes the experimental results for the Linux platform. All the memory snapshots were copied to an external USB 2.0 hard drive without direct I/O. The uncertainty values for the GNU `dd` and `dc3dd` tools are $U_{GNU\texttt{dd}} = 24.4\% \pm 0.74\%$ and $U_{\texttt{dc3dd}} = 25.3\% \pm 0.10\%$, respectively. Note that the uncertainty values and collection times for the two tools are very similar.

Table 4. Tool uncertainty for a virtual Linux system (direct I/O).

Tool	S	MP	μ_{dmp}			σ_{dmp}			T_{coll}	U_{tot}
	[n°]	[n°]	[n°]	[MB]	[%]	[n°]	[MB]	[%]	[sec]	[%]
GNU `dd`	10	131,072	5,950	23.24	4.53	121	0.47	0.09	35	4.53 ± 0.09
`dc3dd`	10	131,072	5,035	19.67	3.84	104	0.41	0.08	33	3.84 ± 0.08

Table 4 shows the results obtained for the virtual Linux system with direct I/O communications (USB transfer with `DIRECT_IO` and user mode buffer size of 50 KB). This was accomplished via the `oflag=direct` parameter using the command:

```
dd if=/dev/mem of=/media/SIGMA/img1/dump01.dd oflag=direct bs=50K
```

The parameter `bs` sets the size of the buffer in user mode, which avoids the use of the Linux kernel page cache [2]. The snapshots were copied to the external USB storage device with an average time of 35 seconds (average transfer speed of 15 MB/s). The estimated footprints of the

Table 5. GNU `dd` tool uncertainty for different buffer sizes.

Uncertainty (U) [%]	**Buffer Size [KB]**	**Collection Time [sec]**	**Transfer Speed [MB/sec]**
6.9 ± 0.09	5	235	2.3
5.3 ± 0.11	10	107	5.0
6.2 ± 0.06	15	93	5.8
4.2 ± 0.07	30	60	8.9
3.9 ± 0.09	60	35	15.1
3.7 ± 0.08	100	33	16.0
4.6 ± 0.13	500	29	18.0
4.1 ± 0.14	1,024	28	19.2

GNU `dd` and `dc3dd` collection tools are $U_{GNU\texttt{dd}} = 4.53\% \pm 0.09\%$ and $U_{\texttt{dc3dd}} = 3.84\% \pm 0.08\%$, respectively. Note that the uncertainty for the two collection tools is reduced by almost five times. Avoiding the page cache also provides better memory coherence.

It is important to be aware of how much the user mode buffer size influences the uncertainty. For this purpose, a set of statistically meaningful, albeit small, memory snapshots were collected with direct I/O communications using different user mode buffer sizes. Table 5 presents the uncertainty values for the GNU `dd` tool (based on five snapshots) for buffer sizes ranging from 5 KB to 1,024 KB. Note that the `bs` parameter specifies equal input and output buffers for the reading and writing phases for the GNU `dd` and `dc3dd` tools. The lowest uncertainty value is obtained with the buffer set to 100 KB ($U_{GNU\texttt{dd}} = 3.7\% \pm 0.08\%$) with a collection time of 33 seconds and transfer speed of 16.0 MB/s.

In another experiment, the effect of LAN communications on memory snapshots was determined by transferring the snapshots to a remote Windows system via a network link. This was accomplished using the command:

```
dd if=/dev/mem | nc 192.168.1.12 10000
```

Table 6. Tool uncertainty with LAN communications.

Tool	**S**	**MP**		μ_{dmp}			σ_{dmp}		T_{coll}	U_{tot}
	[n°]	[n°]	[n°]	[MB]	[%]	[n°]	[MB]	[%]	[sec]	[%]
GNU `dd`	10	131,072	6,331	24.73	4.83	393	1.53	0.30	40	4.83 ± 0.30
`dc3dd`	10	131,072	6,034	23.57	4.60	248	0.97	0.19	39	4.60 ± 0.19

Table 6 shows the uncertainty values in the case of LAN communications: $U_{GNU\texttt{dd}} = 4.83\% \pm 0.30\%$ and $U_{\texttt{dc3dd}} = 4.60\% \pm 0.19\%$. The

Table 7. Tool uncertainty under heavy RAM load without direct I/O.

Tool	**S**	**MP**		μ_{dmp}			σ_{dmp}		T_{coll}	U_{tot}
	[n°]	**[n°]**	**[n°]**	**[MB]**	**[%]**	**[n°]**	**[MB]**	**[%]**	**[sec]**	**[%]**
GNU `dd`	10	131,072	121,134	437.18	92.4	133	0.52	0.10	25	92.4 ± 0.10
`dc3dd`	10	131,072	118,036	461.08	90.0	248	0.97	0.19	26	90.0 ± 0.19

memory snapshots were copied to the host system using `netcat` with an average time of 40 seconds. The results in Table 6 demonstrate that the uncertainty is comparable with that obtained for a virtual system with direct I/O.

Experiments were also conducted to verify how much the volatile memory load (i.e., percentage of volatile memory being used by running processes) impacts collection by the GNU `dd` tool. This can occur when malware consumes large amounts of RAM, possibly impacting the volatile memory collection.

To conduct the experiments, a custom script was created to progressively allocate memory in 1 MB blocks during the collection process. Initially, all the memory snapshots were copied without direct I/O communications; direct I/O was set in a subsequent series of experiments. This enabled us to verify the net effect of direct communications on the memory snapshot uncertainty.

Table 7 shows the results under heavy RAM loads without direct I/O communications. The uncertainty for the GNU `dd` tool is $U_{GNU\texttt{dd}} = 92.4 \pm 0.10\%$ or $U_{GNU\texttt{dd}} = 437.18 \pm 0.52$ MB. Also, the memory snapshots were copied to the host system via the USB 2.0 link with an average time of 25 seconds. Similar results were obtained for the `dc3dd` tool.

Table 8. Tool uncertainty under heavy RAM load with direct I/O.

Tool	**S**	**MP**		μ_{dmp}			σ_{dmp}		T_{coll}	U_{tot}
	[n°]	**[n°]**	**[n°]**	**[MB]**	**[%]**	**[n°]**	**[MB]**	**[%]**	**[sec]**	**[%]**
GNU `dd`	10	131,072	82,652	322.9	63.06	534	2.09	0.41	31	63.06 ± 0.41
`dc3dd`	10	131,072	82,302	321.5	62.80	456	1.78	0.34	30	62.80 ± 0.34

Table 8 shows the experimental results under heavy RAM loads with direct I/O communications. The uncertainty values for the GNU `dd` tool is $U_{GNU\texttt{dd}} = 63.06 \pm 0.41\%$ and $U_{GNU\texttt{dd}} = 322.9 \pm 2.09$ MB. The memory snapshots were copied to the host system via USB in an average time of 31 seconds. Similar results were obtained for the `dc3dd` tool.

Note that the difference between the uncertainty values in the two cases is 29.4%, which is due to the Linux kernel page cache. This result is not surprising because we have already determined the net effect of the page cache. As a consequence, it is necessary to use the direct I/O communications mode when performing a live memory acquisition on a Linux system.

5.3 Practical Consequences of Uncertainty

The experimental results show that the ordinary uncertainty for the collection tools applied to a Windows system is high, especially when using a USB connection to transfer memory snapshots. The notable exception is the `win32dd` tool, which affects the volatile memory less than the other tools. Also, memory dumps are more coherent (i.e., have lower uncertainty) when a LAN connection is used to transfer snapshots.

The net effect of a high uncertainty in the memory snapshots is that some memory pages (even processes) can be swapped out to a dedicated disk partition or a specific set of files depending on the operating system [2]. In some situations, especially when many processes that require plenty of RAM are running, there could be an "out of memory" event, which forces the operating system to kill a running process, usually the last one that was started. Such a critical situation can produce an evidence "leak" or even prevent the collection tool from running.

Increased usage of the page cache in Windows and Linux systems can also cause process pages to be swapped out. If this occurs during a digital forensic investigation, it is necessary to copy the volatile memory (RAM contents) as well as the swap area in order to recover all the process pages. In the case of Windows [8, 18], it is possible to recover almost all the pages related to a running process, even if some pages have been swapped out.

In a Linux system, it is not clear if some processes are deleted or altered when a collection tool runs with direct I/O. Nevertheless, it is preferable to use the direct I/O mode in digital forensic investigations because the volatile memory undergoes less changes.

6. Conclusions

Every digital forensic practitioner should be aware of how much the volatile memory is affected when a forensic tool is used in a live investigation. In particular, it is important to know and possibly predict the extent of the memory perturbation or uncertainty. Experimental results demonstrate that our methodology for evaluating the uncertainty of collection tools is simple and effective, and is applicable to Windows and

Linux systems. The methodology may also be used to evaluate the extent to which a generic forensic tool or toolkit perturbs volatile memory. Although the measurements are limited by the uncertainty of the probing tool, it is still possible to obtain a range of uncertainty for a forensic procedure that affects the volatile memory more than the probing tool itself.

References

[1] D. Bem, Computer forensic analysis in a virtual environment, *International Journal of Digital Evidence*, vol. 6(2), 2007.

[2] D. Bovet and M. Cesati, *Understanding the Linux Kernel*, O'Reilly, Sebastopol, California, 2006.

[3] E. Casey, Error, uncertainty and loss in digital evidence, *International Journal of Digital Evidence*, vol. 1(2), 2002.

[4] e-fense, Helix3 Enterprise, Washington, DC (www.e-fense.com/helix), 2009.

[5] G. Garner, Forensic Acquisition Utilities (www.gmgsystemsinc.com/fau), 2009.

[6] J. Halpern, *Reasoning about Uncertainty*, MIT Press, Cambridge, Massachusetts, 2005.

[7] HBGary, FastDump Pro, Sacramento, California (www.hbgary.com/products-services/fastdump-pro).

[8] J. Kornblum, Using every part of the buffalo in Windows memory analysis, *Digital Investigation*, vol. 4(1), pp. 24–29, 2007.

[9] D. Lindley, *Understanding Uncertainty*, John Wiley, Hoboken, New Jersey, 2006.

[10] C. Malin, E. Casey and J. Aquilina, *Malware Forensics: Investigating and Analyzing Malicious Code*, Syngress, Burlington, Massachusetts, 2008.

[11] Mandiant, Memoryze, Washington, DC (www.mandiant.com/software/memoryze.htm).

[12] ManTech, Memory DD, Vienna, Virginia (cybersolutions.mantech.com/products.htm).

[13] M. McDougal, Windows Forensic Toolchest (WFT) (www.foolmoon.net/security/wft), 2005.

[14] National Institute of Standards and Technology, The NIST Reference on Constants, Units and Uncertainty, Gaithersburg, Maryland, 2006.

[15] M. Oliveira, R. Redin, L. Carro, L. da Cunha Lamb and F. Wagner, Software quality metrics and their impact on embedded software, *Proceedings of the Fifth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, pp. 68–77, 2008.

[16] P. Rubin, D. MacKenzie and S. Kemp, GNU `dd` (www.gnu.org/software/coreutils).

[17] P. Rubin, D. MacKenzie, S. Kemp, J. Kornblum and A. Medico, `dc3dd` (dc3dd.sourceforge.net).

[18] A. Savoldi and P. Gubian, Blurriness in live forensics: An introduction, *Proceedings of the Third International Conference on Information Security and Assurance*, pp. 119–126, 2009.

[19] A. Savoldi and P. Gubian, Volatile memory collection and analysis for Windows mission-critical computer systems, *International Journal of Digital Crime and Forensics*, vol. 1(3), pp. 42–61, 2009.

[20] M. Suiche, `win32dd` (win32dd.msuiche.net).

Chapter 13

IDENTIFYING VOLATILE DATA FROM MULTIPLE MEMORY DUMPS IN LIVE FORENSICS

Frank Law, Patrick Chan, Siu-Ming Yiu, Benjamin Tang, Pierre Lai, Kam-Pui Chow, Ricci Ieong, Michael Kwan, Wing-Kai Hon and Lucas Hui

Abstract One of the core components of live forensics is to collect and analyze volatile memory data. Since the dynamic analysis of memory is not possible, most live forensic approaches focus on analyzing a single snapshot of a memory dump. Analyzing a single memory dump raises questions about evidence reliability; consequently, a natural extension is to study data from multiple memory dumps. Also important is the need to differentiate static data from dynamic data in the memory dumps; this enables investigators to link evidence based on memory structures and to determine if the evidence is found in a consistent area or a dynamic memory buffer, providing greater confidence in the reliability of the evidence. This paper proposes an indexing data structure for analyzing pages from multiple memory dumps in order to identify static and dynamic pages.

Keywords: Live forensics, volatile data, memory analysis

1. Introduction

In recent years, there has been a growing need for live forensic techniques and tools [10]. Best practices have been specified to ensure that acquisition methods minimize the impact on volatile system memory and that relevant evidentiary data can be extracted from a memory dump [3, 12, 16]. However, most approaches focus only on a single snapshot of system memory, which has several drawbacks.

One of the most significant drawbacks is that dynamic activities cannot be detected and analyzed using a single snapshot of memory. Examples of dynamic activities include P2P file sharing and botnet communications. Investigators can uncover valuable evidence by analyzing the memory of processes corresponding to these activities. However, one of the major challenges in undertaking such an analysis is that memory allocation to processes is highly dependent on the system. Different programs often use different memory addressing schemes, causing great discrepancies in memory data structures. Consequently, data recovered from different portions of memory may require different interpretations. By classifying memory into static and dynamic regions, and mapping these regions to logical processes or files, investigators may be able to link evidence found in different portions of different dumps.

Using a single memory snapshot can bring into question the reliability of the extracted evidence and the veracity of the corresponding analysis. Multiple consecutive snapshots of memory can help address this issue. If the memory regions can be classified as static and dynamic, then evidence found in the static area would exist in several consecutive dumps and the integrity of the static evidence can be verified. Conversely, evidence found in a dynamic area can be correlated with other evidence by linking it to the corresponding logical process or file.

The complete analysis of multiple consecutive memory dumps is a challenging, multifaceted problem. This paper addresses one component of the larger problem: Given multiple consecutive snapshots of memory from a Windows system, how can static regions be efficiently differentiated from dynamic regions?

Solving this problem can help understand the memory structure of processes, but the solution is not as simple as it might appear. The memory dumps to be analyzed are huge (2 GB or more). Reading a single 2 GB dump takes more than 30 minutes; processing multiple dumps can take days. Additionally, the definition of the term "static" varies according to the memory processes analyzed and the time interval between consecutive dumps. Thus, the method should be flexible enough to answer which pages are identical in X consecutive dumps and which pages vary in all X consecutive dumps for different values of X. A brute force approach to scanning all the memory dumps for multiple values X would take far too much time.

In this paper, we assume that application programs employ a static memory allocation algorithm for their stack frames. Since memory is divided into pages (4 KB/page on a Windows platform), we adopt a hashing algorithm to assist in the identification of changes between memory pages. Then, we create an indexing data structure to store hash values so that by scanning the dumps just once, it is possible to efficiently answer which pages are static (identical) or dynamic (different) in X consecutive dumps for different values of X.

2. Related Work

Given the limitations of conventional digital forensic techniques for dealing with volatile memory, the acquisition and analysis of machine memory are currently hot topics of research [5, 9]. Substantial research has focused on tools that can acquire memory images without altering memory content [7, 10, 11, 15]. However, the dynamic nature of memory means that obtaining a complete and consistent perspective of memory is impossible without taking multiple memory snapshots. In addition to problems posed by memory fragmentation, the analysis of data is complicated by the fact that memory structures vary considerably for different systems [16].

Microsoft Windows is the most common operating system encountered by digital forensic examiners. Much research has been directed at extracting relevant data from live Windows systems [4]. However, the closed source nature of Windows makes it difficult to verify the results, potentially increasing the likelihood of challenges when the evidence is presented in court.

Instead of focusing on data acquisition [4, 6] and memory object reconstruction [3, 12, 14], Arasteh and Debbabi [1] investigated the process memory stack. By analyzing the stacking mechanism in Windows memory, they were able to discover the partial execution history of a program in the memory process stack, which can be of value in forensic investigations. Chow, *et al.* [8] pointed out the possibility of differentiating static data corresponding to a UNIX memory process in order to identify useful data from inconsistent data in a memory dump. Balakrishnan and Reps [2, 13] analyzed memory accesses by x86 executables and proved the viability of distinguishing various regions of memory data created by executables. The heap and global data regions are areas where persistent data can be found. Balakrishnan and Reps also demonstrated the importance of understanding memory structures and of classifying static and dynamic data in memory dumps. In the following, we examine this issue in more detail in the context of forensic investigations.

3. Methodology

The problem studied in this paper can be stated as follows. Given K consecutive memory dumps, each containing N pages, and a sequence of m queries related to which pages have identical contents (static pages) in X consecutive dumps or which pages have different contents (dynamic

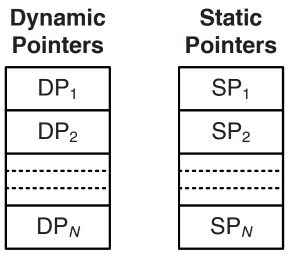


Figure 1. Data structure for the indexing approach.

pages) in all X consecutive dumps, then identify the page numbers and their corresponding dumps. Note that the value of X can be different in different queries in problem specification.

In the following, we assume that a hash value (e.g., MD5 or SHA-1) is computed and stored for each page. Comparing the hash values for a given page in different dumps identifies if their contents are identical or different.

3.1 Brute Force Approach

First, we describe a brute force approach in which the memory dumps taken at different times are read once for each query. We show how this approach is used to identify dynamic pages; static pages are identified in a similar manner.

The first page of each of the K dumps is read, and their hash values are computed and stored in an array. The array is scanned once: if X or more consecutive entries in the array are different, then the page is a dynamic page. This procedure is repeated for the other pages.

The brute force approach involves significant overhead for each query. In particular, considerable I/O time is expended to read all the memory dumps repeatedly. This problem is addressed in our indexing approach.

3.2 Indexing Approach

The indexing approach involves reading all the memory dumps only once and building a data structure so that queries can be answered efficiently. The data structure is essentially an array of linked lists as shown in Figure 1. Two linked lists are maintained for each of the N pages in the memory, the dynamic list and the static list. For Page i, the dynamic pointer DP_i and the static pointer SP_i point to the dynamic list and static list, respectively. The dynamic list pointed to by DP_i helps locate groups of dynamic dumps corresponding to Page i. Similarly, the static list pointed to by SP_i helps locate groups of static dumps for Page

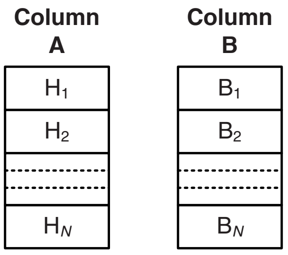


Figure 2. Working array for building the data structure.

i. Therefore, each node in a list – whether static or dynamic – denotes a group of dumps. In addition to pointers to the next node and the previous node, a count and starting position are stored for each node. The count gives the number of dumps in the group and the starting position gives the position of the first dump in the group. All the lists are sorted in descending order based on the count.

We now show how this data structure can handle a query for dynamic pages; the query for static pages is handled in a similar manner. The first node pointed to by DP_i is scanned once for each i from 1 to N. For each node with count C and starting position S, if C is greater than or equal to X, then Page i in Dump S with length C is a dynamic page. This procedure is repeated for the next node pointed to by DP_i until a node with count less than X is reached. At this stage, the procedure continues with the next dynamic pointer DP_{i+1} until the last dynamic pointer DP_N is processed.

A 2-D temporary working array is required to create the data structure (Figure 2). As Dump 1 is read, the hash value of each Page i in the dump is stored as H_i. When Dump 2 is read, the hash value of each Page i in this dump is compared with H_i. If the hash values are identical, then B_i is set to False, a new node is created with count equal to 2 and starting position equal to 1, and the static pointer SP_i is set to point to the new node. Otherwise, B_i is set to True, a new node is created with count equal to 2 and starting position equal to 1, and the dynamic pointer DP_i is set to point to the new node.

For Dump 3 onwards, the following steps are performed. Read Dump j. For each Page i in Dump j with $H_i^{'}$, compare $H_i^{'}$ with H_i. There are four cases.

- **Case 1:** If $H_i^{'}$ equals H_i and B_i is True, then create a new node with count equal to 2 and starting position equal to $j-1$, and insert the new node at the head of the list pointed to by SP_i. Set B_i to False.

- **Case 2:** If $H_i^{'}$ equals H_i and B_i is False, then increase the count of the head node of the list pointed to by SP_i by 1.

- **Case 3:** If $H_i^{'}$ is not equal to H_i and B_i is False, then create a new node with count equal to 2 and starting position equal to $j-1$, and insert the new node at the head of the list pointed to by DP_i. Set H_i to $H_i{'}$ and B_i to True.

- **Case 4:** If $H_i{'}$ is not equal to H_i and B_i is True, then increase the count of the head node of the list pointed to by DP_i by 1. Set H_i to $H_i^{'}$.

Finally, sort all the lists in descending order based on the count.

3.3 Time and Space Complexity

This section conducts an analysis of the time and space complexity of the two approaches. Let K be the number of memory dumps, N be the number of pages in each dump and m be the number of queries to be answered.

In the case of the brute force approach, an array of just K entries is needed; therefore, the space complexity is $O(K)$. For each query, all the memory dumps have to be scanned once; therefore, the time complexity for each query is $O(N \times K)$. Since m is the number of queries to be answered, the overall time complexity is $O(mNK)$. Note that all these operations involve I/O as the memory dump has to be read each time a query is answered. Consequently, the brute force approach requires $O(mNK/B)$ I/O operations where B is the I/O block size.

In the case of the indexing approach, for K memory dumps with N pages per dump, a 2-D working array with 2 columns and N rows is needed; therefore, the space complexity for the working array is $O(N)$. The data structure requires two lists per page, a total of $2N$ lists. In the worst-case situation, where the dumps are the same for every two pages, there are about $K/2$ nodes in each list. Thus, there are $N \times K$ nodes for the entire data structure and the space complexity is, therefore, equal to $O(N \times K)$.

Next, we consider the time complexity of the indexing approach. Building the data structure requires each memory dump to be scanned once, and this is the only step involving I/O operations. The time complexity for this step is $O(NK/B)$ I/O operations where B is the I/O block size. Next, all the linked lists have to be sorted; since there are at most $K/2$ nodes in each list, the time complexity for this step is $N \times 2 \times (K/2)\log(K/2)$. Thus, the overall time complexity is $O(N \times K \log K)$.

To answer a query, it is necessary to scan the first node in all the dynamic lists or in all the static lists. If L is the number of nodes that have to be scanned, then the time complexity is $O(L)$ for each query (these operations are much faster because no I/O operations are involved). The total time complexity for handling m queries is $O(NK \log K + mL)$. Note that in real-world scenarios, where L is much less than $N \times K$, the indexing approach is much faster than the brute force approach. In summary, the overall time complexity consists of two parts: $O(NK/B)$ I/O operations to read the memory dumps and $O(NK \log K + mL)$ RAM operations to handle m queries.

4. Experimental Results

This section compares the performance of the two approaches in terms of running time and running space. The two approaches were implemented in C++ code and executed on a Core2Duo P8400 2.26GHz computer with 4 GB RAM. The memory dumps used in the experiments were obtained from a video playback program, which was playing a video when the memory dumps were acquired. Obviously, memory dumps taken under such conditions are very dynamic.

The experiments involved three rounds. In Round 1, ten memory dumps were taken of the video playback program; each dump was about 59,660 KB. The two approaches were executed to identify pages with K consecutive dynamic dumps ($K = 5..10$). Therefore, a total of six queries were issued.

Table 1. Running times (Round 1).

	K = 5	**K = 6**	**K = 7**	**K = 8**	**K = 9**	**K = 10**
Brute Force	6.27 min	3.87 min	3.84 min	3.88 min	5.64 min	3.77 min
Indexing	2.68 min	1.13 min	1.03 min	1.06 min	1.07 min	1.04 min

Table 1 presents the running times obtained for the two approaches for various values of K. Note that the running times were measured using the C++ internal time library. The running space, which was obtained using the process manager, was 496 KB for the brute force approach compared with 504 KB for the indexing approach.

In Round 2, ten additional dumps were taken. As in Round 1, each dump was about 59,660 KB. Table 2 presents the running times for six values of K ($K = 15..20$). The running space was 584 KB for the brute force approach compared with 500 KB for the indexing approach.

Table 2. Running times (Round 2).

	K = 15	K = 16	K = 17	K = 18	K = 19	K = 20
Brute Force	10.03 min	12.94 min	13.82 min	15.74 min	17.70 min	19.94 min
Indexing	20.86 min	41.89 sec	40.89 sec	40.77 sec	40.83 sec	40.82 sec

Table 3. Running times (Round 3).

	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10
Brute Force	154 min	160 min	166 min	170 min	162 min	180 min
Indexing	260 min	15 min	15 min	15 min	15 min	15 min

Round 3 involved ten larger memory dumps (about 2 GB each) to simulate the analysis of the entire memory dump of a computer. The two approaches were then used to identify pages with K consecutive dynamic dumps ($K = 5..10$). Table 3 presents the running times. The running space was 668 KB for the brute force approach compared with 516 KB for the indexing approach.

The experimental results show that the indexing approach requires significantly less time than the brute force approach except for $K = 15$ in Round 2 and $K = 5$ in Round 3. This is because, in order to answer the first query, the indexing approach has to build the data structure, which takes some time. However, the indexing approach uses the same data structure for subsequent queries. Consequently, in the later runs, the indexing approach is much faster than the brute force approach. Finally, as far as the running space is concerned, the two approaches require approximately the same amount of memory.

5. Conclusions

The indexing approach is designed to identify static and dynamic pages in multiple consecutive memory dumps. It is much faster than the brute force approach and uses about the same amount of memory

It is important to note that our approach is just the first step in the analysis of volatile data using multiple memory snapshots. Nevertheless, the approach could be refined and augmented to help identify static and dynamic data corresponding to a process in memory, track the address of the stack area, identify data that is consistently retained within memory, and link evidence related to the dynamic activities of users. These are all challenging research problems in their own right and are deserving of further investigation.

References

[1] A. Arasteh and M. Debbabi, Forensic memory analysis: From stack and code to execution history, *Digital Investigation*, vol. 4(S), pp. S114–S125, 2007.

[2] G. Balakrishnan and T. Reps, Analyzing memory accesses in x86 executables, *Proceedings of the Thirteenth International Conference on Compiler Construction*, pp. 5–23, 2004.

[3] Bugcheck, GREPEXEC: Grepping executive objects from pool memory (www.uninformed.org/?v=4&a=2&t=pdf), 2006.

[4] M. Burdach, An introduction to Windows memory forensics (forens ic.seccure.net/pdf/introduction_to_windows_memory_forensic.pdf), 2005.

[5] M. Burdach, Digital forensics of the physical memory (forensic.sec cure.net/pdf/mburdach_digital_forensics_of_physical_memory.pdf), 2005.

[6] M. Burdach, Windows Memory Forensic Toolkit (forensic.seccure .net), 2007.

[7] B. Carrier and J. Grand, A hardware-based memory acquisition procedure for digital investigations, *Digital Investigation*, vol. 1(1), pp. 50–60, 2004.

[8] K. Chow, F. Law, M. Kwan and P. Lai, Consistency issues in live systems forensics, *Proceedings of the International Workshop on Forensics for Future Generation Communication Environments*, pp. 136–140, 2007.

[9] D. Farmer and W. Venema, *Forensic Discovery*, Addison-Wesley, New York, 2005.

[10] G. Garcia, Forensic physical memory analysis: Overview of tools and techniques, Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology, Helsinki, Finland (www.tml .tkk.fi/Publications/C/25/papers/Limongarcia_final.pdf), 2007.

[11] E. Huebner, D. Bem, F. Henskens and M. Wallis, Persistent systems techniques in forensic acquisition of memory, *Digital Investigation*, vol. 4(3-4), pp. 129–137, 2007.

[12] N. Petroni, A. Walters, T. Fraser and W. Arbaugh, FATKit: A framework for the extraction and analysis of digital forensic data from volatile system memory, *Digital Investigation*, vol. 3(4), pp. 197–210, 2006.

[13] T. Reps and G. Balakrishnan, Improved memory-access analysis for x86 executables, *Proceedings of the Seventeenth International Conference on Compiler Construction*, pp. 16–35, 2008.

[14] A. Schuster, Searching for processes and threads in Microsoft Windows memory dumps, *Digital Investigation*, vol. 3(S1), pp. S10–S16, 2006.

[15] I. Sutherland, J. Evans, T. Tryfonas and A. Blyth, Acquiring volatile operating system data: Tools and techniques, *ACM SIGOPS Operating Systems Review*, vol. 42(3), pp. 65–73, 2008.

[16] R. van Baar, W. Alink and A. van Ballegooij, Forensic memory analysis: Files mapped in memory, *Digital Investigation*, vol. 5(S), pp. S52–S57, 2008.

Chapter 14

A COMPILED MEMORY ANALYSIS TOOL

James Okolica and Gilbert Peterson

Abstract The analysis of computer memory is becoming increasingly important in digital forensic investigations. Volatile memory analysis can provide valuable indicators on what to search for on a hard drive, help recover passwords to encrypted hard drives and possibly refute defense claims that criminal activity was the result of a malware infection. Historically, digital forensic investigators have performed live response by executing multiple utilities. However, using a single tool to capture and analyze computer memory is more efficient and has less impact on the system state (potential evidence). This paper describes CMAT, a self-contained tool that extracts forensic information from a memory dump and presents it in a format that is suitable for further analysis. A comparison of the results obtained with utilities that are commonly employed in live response demonstrates that CMAT provides similar information and identifies malware that is missed by the utilities.

Keywords: Live response, memory analysis, rootkit detection

1. Introduction

When a large enterprise responds to a cyber security incident, it may not be feasible to perform the recommended disconnection and imaging process to gather forensic evidence [13], especially in the case of servers that are critical to business operations. According to CNET [19], Amazon lost $29,000 in revenue per minute during its June 2008 outage. In addition, there have been several recent cases where defendants were found innocent or guilty based on the live response information retrieved at the time of the incident [2]. Clearly, tools that can take quick snapshots of affected computers while minimizing the impact on business operations and preserving evidence are vital to digital forensic investigations. Since many of the items of interest in a live response are maintained in computer memory rather than the hard drive, a minimally intrusive live response would involve the analysis of a live memory capture.

Three steps are involved in developing a minimally intrusive standalone forensic tool for memory: (i) a front-end memory dump routine that directly accesses physical memory; (ii) a memory analysis tool that extracts environmental and activity information from the memory dump; and (iii) a synthesis tool that correlates the environmental information to provide a human-understandable narrative of what occurred on the machine.

This paper focuses on the second step: developing a compiled memory analysis tool (CMAT) that extracts environmental and activity information from a memory dump. CMAT parses a memory dump to find active, inactive and hidden processes as well as system registry information. It then compiles live response forensic information from these processes and registry files and assembles it into a format suitable for data correlation. CMAT is tested against several utilities that extract forensic information from a live system. Experimental results indicate that CMAT provides comparable information as traditional live response utilities, and also uncovers malware that these tools miss.

2. Background

Performing a live response allows the capture of forensic information that disappears after the computer is turned off. The benefits of a live response include: gathering clues to better focus a search of the hard drive [24]; retrieving decryption/encryption keys for an encrypted hard drive [24]; and defeating the Trojan horse defense of “I didn’t do it, it was the malware installed on my machine” [2]. Valuable live response information that resides in memory includes [4, 5, 10, 24]:

- System date and time
- Logged in users and their authorization credentials
- Network information, connections and status
- Process information, memory and process-to-port mappings
- Clipboard contents
- Command history
- Services and driver information
- Open files, registry keys and hard disk images

One of the arguments against live response is that the evidence capture process modifies the machine state. Several techniques have been proposed to address this issue [1, 3, 9, 22]. These techniques attempt to capture all the memory instead of specific pieces of forensic information. This helps answer the immediate forensic questions and also provides a means to answer additional questions that may come up later in the investigation. The alternatives include hardware-based memory acquisition [1, 3], virtual machines, hibernation files [22] and operating system patches [9].

Hardware-based memory acquisition techniques circumvent the reliance on the operating system by using hardware that interacts directly with memory. These techniques involve the use of custom hardware [3] or the FireWire interface [1] for direct memory access. But the two techniques suffer from the "northbridge exploit" [16]. Since all peripheral devices use the northbridge to interface with the CPU and main memory, malware can be loaded onto the northbridge to subvert memory access by a peripheral while leaving the CPU and operating system untouched. This enables the malware to remain undetected.

The northbridge exploit can be contained by constraining memory capture techniques to the CPU and operating system. Virtual machines provide an easy method for dumping memory. When an incident occurs, a forensic investigator takes control of the actual machine, suspends the virtual machine, and then dumps the memory in the virtual machine. However, one problem with this method is that implementing such a policy within a large organization incurs high costs in terms of time and money. Moreover, software virtualization can slow the perceived responsiveness of a computer because every command has to go through an additional level of abstraction.

Hibernation files offer an alternative memory capture method that relies on the operating system [22]. The process of analyzing hibernation files requires placing the system in the hibernation mode, copying the hibernation files and extracting their contents. However, while the hibernation function is enabled on many laptops, it is disabled by default on most desktops. Desktops can be reconfigured to enable the hibernation mode, but this must be done in advance. Other difficulties include the lack of published descriptions of hibernation file formats and the fact that not everything in memory may be stored.

Libster and Kornblum [9] have proposed an alternative method for acquiring a pristine copy of memory. The method involves modifying the operating system kernel so that a user-defined function key would cause the operating system to immediately suspend, generate a memory dump that it sends over a network to a secure machine, and then resume execution. The difficulty with this method is to ensure that it is deployed for all the disparate systems of a large enterprise prior to an incident.

While a pre-installed operating system patch appears to be an ideal solution in cases where prior access is not possible, the only option available is to initiate a new process to dump system memory. To minimize the impact, the application should limit application program interfaces (APIs) that interface with the operating system and the use of a graphical user interface (GUI). Unfortunately, while many available tools (e.g., [12, 23]) are run from the command line, most use operating system APIs to dump memory. The only exception appears to be Memoryze [11].

Memory analysis tools extract different amounts and types of live response data from a memory dump. The Volatility tool [25] extracts the date and time, running processes, open ports, process-port mappings, strings-process mappings, process-file mappings, process dynamic link libraries (DLLs) and open connections. Additional plug-ins are available to enhance the functionality of Volatility, including the ability to extract registry information. However, Volatility suffers from the limitation that it does not extract port information for all Windows operating system service pack and patch configurations.

Comprehensive open source memory analysis tools written in compiled languages such as C/C++ are not available for Windows XP systems. Betz [21] has developed a tool that finds processes and threads in memory and extracts some information, but it is limited to Windows 2000 systems and does not support the analysis of registry files, network activity and process creators. PTfinder [17] also searches through memory for processes and threads, and handles memory captures from Windows XP systems. However, like Betz's tool, it does not support the analysis of registry files [7], network activity and process creators [8].

3. Compiled Memory Analysis Tool

The compiled memory analysis tool (CMAT) is a C++ command line utility that runs on Linux and Windows operating systems. It parses a Windows XP memory dump to obtain information on user accounts, the Windows registry and running processes. CMAT provides system, process, registry, open ports and user information in a standalone tool that runs without API calls or high-level language interpreters. CMAT operates in two passes. During the first pass, CMAT searches the memory dump file for processes, threads and registry hives. In the second pass, CMAT organizes the process, thread and registry entries it has found and produces output in a format similar to that produced by common live response tools.

The Windows kernel incorporates data structures needed for the operating system to function. These data structures are useful for extracting live response information. For example, processes are stored in memory as _EPROCESS structures. Potential process structures can be found by searching for strings formatted as a _DISPATCH_HEADER of a _KPROCESS. In the case of Windows XP SP2, this corresponds to a _DISPATCH_HEADER of `0x03` and a _KPROCESS size of `0x1b`. After a likely process is found, it is double-checked to ensure that the page directory table base is a valid virtual memory location [15, 18]. If the page directory table does not point to a valid location, then what has been found is a process executable or process data, not a process header. This method finds all processes, including normal active processes, hidden active processes (e.g., processes hidden by a rootkit like FUTo [20]), defunct processes and processes from previous boots that have not been removed from memory. In addition, because the _EPROCESS structure also contains a doubly-linked list of all active processes, processes that were missed during the string search can still be found by traversing the linked list. The scan of the linked list also assists in identifying process that have been disconnected from the list (which is how FUTo hides processes).

CMAT then searches through the memory dump for registry hives. In a manner similar to how it searches for processes, CMAT looks for _CMHIVE structures with a signature in Windows XP of `0xbee0bee0` [7]. Just as a process must have a valid memory address for its page directory table base, a registry hive must have a valid memory address for its base block. CMAT also takes advantage of the doubly-linked _CMHIVE structures to ensure that all the registries are found. CMAT then moves through the default user registry to find relationships between session IDs and user names. In Windows XP, it finds the ProfileList key under *\USER\MICROSOFT\WindowsNT\CurrentVersion* and then makes a record of the session IDs and their associated ProfileImagePaths, providing human-understandable names for the users that were logged in.

As a part of its processing, CMAT translates the virtual addresses stored in Windows kernel data structures into physical memory locations. Windows XP has two levels of indirection when it is run on a 32-bit machine; this is achieved by splitting a virtual address into a page directory index, a page table index and an offset. However, if large (4 MB) pages are used, there is one level of indirection; if physical address extensions are used, there are three levels of indirection. While CMAT can detect if large pages are being used, it requires user assistance in the form of a parameter value to discern if physical address extensions are in use. One method to avoid this is to try alternative virtual-to-physical mappings until the correct mapping is found [25]. Another method is to locate the kernel system variables that store this information [6].

After the process entries and registry hives are found, CMAT either interactively or in the batch mode provides forensic information of interest. This includes general system information (operating system and number of processes) and process information, which includes the user who created the process, full path of the executable, command line used to initiate the process, full paths of loaded DLLs, and the files and registry keys accessed by the process.

4. Test Methodology

The CMAT output was compared with the outputs of five Sysinternals utilities: `psinfo`, `pslist`, `logonsessions`, `handles` and `listdlls` [14]. System information that was examined included the operating system version, number of processors and number of processes. The process information examined included the process creator, files opened, registry keys accessed and modules loaded. The only type of volatile information not examined was network information, including the ports and sockets opened by processes. Also, a comparison was made of the number of distinct DLLs used by ManTech's physical memory dump (`mdd`) utility compared with the number of distinct DLLs used by the Sysinternals utilities.

A Windows XP SP3 system with 2,038 MB RAM was used in the tests. Several application programs were launched on the machine including Internet Explorer, Word, PowerPoint, Visual Studio, Calculator, Kernel Debugger and two command line shells. One of the command line shells was hidden by the FUTo rootkit [20]. The memory dump was collected using ManTech's `mdd` utility (version 1.3), which required 67 seconds.

5. Test Results

The test results in Table 1 demonstrate that CMAT provides the same or equivalent information as the Sysinternals utilities. In addition, CMAT found the process hidden by FUTo while the Sysinternals utilities did not. The Sysinternals utilities failed to associate several processes with the users that started them. Except for the inability of CMAT to provide network information, CMAT's performance in the tests was exemplary.

- **System Information:** CMAT and `psinfo` provide the operating system version, major and minor version, service pack number

Table 1. CMAT vs. Sysinternals utilities.

Information	**Result**	**Correlation**
Operating System Version	Windows XP SP3 v5.1 Build 2600	100%
Processor Count	2	100%
Process Count	56 of 58	97%
User IDs	50 of 57	88%
Loaded Modules	57 of 57	100%
Files	57 of 57	100%
Registry Keys	57 of 57	100%
DLLs Used	15 (CMAT) vs. 48 (Sysinternals)	

and build number. Also, they both provide the number of processors. The `psinfo` utility also provides the processor type and speed along with the video driver used. However, this information appears to have limited forensic use; therefore, this feature was not implemented in CMAT. The one additional piece of information provided by `psinfo` compared with CMAT is if a physical address extension is used (as described above, this information is passed to CMAT as a parameter). A future version of CMAT will address this shortcoming.

- **Processes:** CMAT reported 58 active processes while `pslist` reported 57 processes; 56 of the processes matched (97% equality). The program missing from the CMAT results was `pslist`, which was not running when the memory was dumped. Similarly, the program `mdd_1.3` was missing from the `pslist` result, which makes sense because the memory was already dumped when `pslist` was executed. The other program missing from the `pslist` result was `cmd.exe`, which was hidden by the FUTo rootkit; this demonstrates that a program intentionally hidden by malware can still be found by CMAT. While CMAT provides a single process listing that includes the user who created the process, Sysinternals uses a separate utility (`logonsessions`) for this purpose. Comparison of the CMAT and `logonsessions` results showed that nine processes were not listed by `logonsessions`, including all the processes owned by LocalService and NetworkService, and four processes owned by SystemProfile (including the system and idle processes). Identifying all the system processes is vital to detecting if malicious services were executing on the machine.

- **Loaded Modules:** CMAT and the Sysinternals `listdll` utility provided identical lists of loaded modules (100% equality).

- **Files and Registry Keys:** The lists of files and registry keys provided by CMAT and the Sysinternals `handle.exe` utility matched, except for the temporary files that were opened and closed between the memory dump and the execution of `handle.exe`. These files were listed as having `[RWD]` access. CMAT identified file handles that were actually device handles (e.g., *\Device\KsecDD*), but was unable to print the names of the devices. Similarly, while `handle.exe` summarized the long code for the current user with a succinct HKU, CMAT displayed the entire registry key.

- **Dynamic Link Libraries:** As mentioned above, minimizing the number of loaded modules (DLLs) used when performing a live response is desirable because less evidence on the hard drive is modified. This test compared the numbers of DLLs used by ManTech's `mdd` memory capture tool and the five Sysinternals utilities. To collect the data, `mdd` was executed during the time that each of the five Sysinternals utilities was running. The DLL lists associated with the five Sysinternals utilities were extracted from the memory dumps using CMAT; duplicate DLLs among the five utilities were removed and the results were then tallied. The results show that performing a memory capture instead of a live response significantly reduces the impact on the evidence. In particular, the memory capture used 15 DLLs while the five Sysinternals utilities in total used 48 different DLLs, corresponding to a 69% reduction in system impact.

6. Conclusions

The CMAT live response tool provides comparable information from a memory dump and produces a 69% less impact on system state (potential evidence) than the popular Sysinternals utilities. However, CMAT is unable to extract several pieces of useful forensic information, including the mapping of ports and sockets to processes (provided by the Sysinternals `portmon` utility). The difficulty arises because this information is stored in the data section of the TCP/IP module. While the port and socket data reside in specific locations for different Windows operating system versions, service packs and patch configurations, their locations can change when new patches are installed.

A future version of CMAT will target Microsoft Windows XP operating systems. One difficulty is to correctly map the kernel data structures for different Windows versions (e.g., Vista and Windows 7). This difficulty can be addressed by retrieving the operating system data structures dynamically after the operating system is identified upon parsing the memory dump. Microsoft has recognized the need for such flexibility by providing the data structures (symbol tables) for download in real time to support its kernel debugger. CMAT will attempt to make use of these symbol tables to facilitate memory analysis for different versions of the Windows operating system.

References

[1] A. Boileau, Hit by a bus: Physical access attacks with FireWire (www.storm.net.nz/static/files/ab_firewire_rux2k6-final.pdf), 2006.

[2] S. Brenner, B. Carrier and J. Henninger, The Trojan Horse Defense in Cybercrime Cases, CERIAS Tech Report 2005-15, Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, Indiana, 2005.

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

[4] B. Carrier and J. Grand, A hardware-based memory acquisition procedure for digital investigations, *Digital Investigation*, vol. 1(1), pp. 50–60, 2004.

[5] H. Carvey, *Windows Forensic Analysis*, Syngress, Burlington, Massachusetts, 2007.

[6] B. Dolan-Gavitt, Finding kernel global variables in Windows (mo yix.blogspot.com/2008/04/finding-kernel-global-variables-in.html), April 16, 2008.

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

[8] B. Dolan-Gavitt, Linking processes to users (moyix.blogspot.com /2008/08/linking-processes-to-users.html), August 16, 2008.

[9] E. Libster and J. Kornblum, A proposal for an integrated memory acquisition mechanism, *ACM SIGOPS Operating Systems Review*, vol. 42(3), pp. 14–20, 2008.

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

[11] Mandiant, Memoryze, Washington, DC (www.mandiant.com/soft ware/memoryze.htm).

[12] ManTech, Memory DD, Vienna, Virginia (cybersolutions.mantech .com/products.htm).

[13] National Institite of Justice, Electronic Crime Scene Investigation: An On-the-Scene Reference for First Responders, U.S. Department of Justice, Washington, DC, 2009.

[14] M. Russinovich, Sysinternals Suite, Microsoft Corporation, Redmond, Washington (technet.microsoft.com/en-us/sysinternals/bb 842062.aspx).

[15] M. Russinovich and D. Solomon, *Microsoft Windows Internals*, Microsoft Press, Redmond, Washington, 2005.

[16] J. Rutkowska, Beyond the CPU: Defeating hardware-based RAM acquisition (Part I: AMD case), presented at the *Black Hat DC 2007 Conference* (www.first.org/conference/2007/papers/rutkowska-joa nna-slides.pdf), 2007.

[17] A. Schuster, PTfinder (version 0.2.00), Bonn, Germany (compu ter.forensikblog.de/en/2006/03/ptfinder_0_2_00.html), 2006.

[18] A. Schuster, Searching for processes and threads in Microsoft Windows memory dumps, *Digital Investigation*, vol. 3(S), pp. S10–S16, 2006.

[19] S. Shankland, Amazon suffers U.S. outage on Friday, CNET, San Francisco, California (news.cnet.com/8301-10784_3-9962010-7 .html), June 6, 2008.

[20] P. Silberman, `FUTo`, *Uninformed*, vol. 3 (www.uninformed.org/?v= 3&a=7&t=sumry), January 2006.

[21] SourceForge.net, Memparser (sourceforge.net/projects/mempars er), 2006.

[22] M. Suiche, Sandman Project (sandman.msuiche.net/docs/Sand Man_Project.pdf), 2008.

[23] M. Suiche, `win32dd` (win32dd.msuiche.net).

[24] I. Sutherland, J. Evans, T. Tryfonas and A. Blyth, Acquiring volatile operating system data: Tools and techniques, *ACM SIGOPS Operating Systems Review*, vol. 42(3), pp. 65–73, 2008.

[25] A. Walters and N. Petroni, Volatools: Integrating volatile memory forensics into the digital investigation process, presented at *Blackhat Hat DC 2007 Conference* (www.blackhat.com/presentations/bh-dc-07/Walters/Paper/bh-dc-07-Walters-WP.pdf), 2007.

V

ADVANCED FORENSIC TECHNIQUES

Chapter 15

DATA FINGERPRINTING WITH SIMILARITY DIGESTS

Vassil Roussev

Abstract State-of-the-art techniques for data fingerprinting have been based on randomized feature selection pioneered by Rabin in 1981. This paper proposes a new, statistical approach for selecting fingerprinting features. The approach relies on entropy estimates and a sizeable empirical study to pick out the features that are most likely to be unique to a data object and, therefore, least likely to trigger false positives. The paper also describes the implementation of a tool (`sdhash`) and the results of an evaluation study. The results demonstrate that the approach works consistently across different types of data, and its compact footprint allows for the digests of targets in excess of 1 TB to be queried in memory.

Keywords: Data fingerprinting, similarity digests, fuzzy hashing

1. Introduction

One of the most common tasks early in the investigative process is to identify known content of interest and to exclude known content that is not of interest. This is accomplished by hashing data objects (typically files) and comparing them to a database of known hashes such as NSRL [10]. The limitations of "known file filtering" become apparent when one attempts to find an embedded object (e.g., JPEG image) inside a document or an archive – file-level hashes are useless and block-level hashes barely make a difference. A similar situation arises when analyzing network traces. Ideally, one would like to quickly identify the presence of objects of interest without paying the overhead of reconstructing network connections and extracting entire files.

Another situation where current approaches fall short is the identification of "similar" objects such as different versions of a document, library or executable. The latter is especially important when dealing with online updates that are common in modern software packages – it is impractical to expect reference databases to contain every single variation of a distribution file.

This paper attempts to address the above scenarios and to develop a practical solution that investigators can employ in the field. The method is based on the idea of identifying statistically-improbable features and using them to generate "similarity digests." Unlike cryptographic digests, which support only yes/no query results, similarity digests allow queries to be answered approximately (in the 0 to 100 range), thereby providing a measure of correlation. The method is specifically designed to deal with the problem of false positives. In prior work [15], we have shown that the root cause of false positives is that the underlying data does not contain enough unique features to be reliably identified. Therefore, the method detects and flags situations in which the query does not contain enough characteristic features for a reliable comparison and notifies the investigator. This paper describes a new tool, `sdhash`, that implements the method and presents an evaluation study that demonstrates its effectiveness.

2. Related Work

In the domain of security and authentication, a fingerprint is often synonymous with the message digest produced by a cryptographic hash function. Identical digests (or signatures) for two different objects are considered conclusive proof that the data objects themselves are identical. Digital forensic investigators make wide use of cryptographic hashes such as SHA-1 to ensure the integrity of forensic targets and to identify known content. The Achilles heal of cryptographic hashes is that they (ideally) depend on every bit of the input, which makes them inherently fragile and unsuited for similarity detection.

2.1 Rabin Fingerprinting

The idea of generating a more flexible and robust fingerprint for binary data was proposed by Rabin in 1981 [13]. Since then, considerable research has focused on developing ever more sophisticated fingerprinting techniques, but Rabin's basic idea has carried over with relatively small variations. We limit our discussion to the essential ideas. Interested readers are referred to [16] for a detailed survey of hashing and fingerprinting techniques.

Rabin's scheme is based on random polynomials and its original purpose was "to produce a very simple real-time string matching algorithm and a procedure for securing files against unauthorized changes" [13]. A Rabin fingerprint can be viewed as a checksum with low, quantifiable collision probabilities that can be used to efficiently detect identical objects. In the 1990s, there was a renewed interest in Rabin's work in the context of finding similar objects, with an emphasis on text. Manber [8] used it in the `sif` tool for Unix to quantify similarities among text files; Brin and colleagues [2] used it in a copy-detection scheme [2]; Broder, *et al.* [3] applied it to find syntactic similarities in web pages.

The basic idea, which is referred to as anchoring, chunking or shingling, is to use a sliding Rabin fingerprint over a fixed-size window that splits data into pieces. A hash value h is computed for every window of size w. The value is divided by a constant c and the remainder is compared with another constant m. If the two values are equal (i.e., $m \equiv h \bmod c$), then the data in the window is declared as the beginning of a chunk (anchor) and the sliding window is moved one position. This process is continued until the end of the data is reached. For convenience, the value of c is typically a power of two ($c = 2^k$) and m is a fixed number between zero and $c - 1$. Once the baseline anchoring is determined, it can be used in a number of ways to select characteristic features. For example, the chunks in between anchors can be chosen as features. Alternatively, the l bytes starting at the anchor positions may be chosen as features, or multiple nested features may be employed.

Note that, while shingling schemes pick a randomized sample of features, they are deterministic, i.e., given the same inputs, produce the same features. Also, they are locally sensitive in that the determination of an anchor point depends only on the previous w bytes of input, where w could be as small as a few bytes. This property can be used to solve the fragility problem in traditional file- and block-based hashing.

Consider two versions of the same document. One document can be viewed as being derived from the other by inserting and deleting characters. For example, an HTML page can be converted to plain text by removing all the HTML tags. Clearly, this would modify a number of features, but the chunks of unformatted text would remain intact and produce some of the original features, permitting the two versions of the document to be automatically correlated. For the actual feature comparison, the hash values of the selected features are stored and used as a space-efficient representation of a "fingerprint."

2.2 Fuzzy Hashing

Kornblum [7] was among the first researchers to propose the use of a generic fuzzy hashing scheme for forensic purposes. His `ssdeep` tool generates string hashes of up to 80 bytes that are the concatenations of 6-bit piecewise hashes. The result is treated as a string and is compared with other hashes on the basis of edit distance – a measure of how many different character insert/delete operations are necessary to transform one string into the other. While `ssdeep` has gained some popularity, the fixed-size hash it produces quickly loses granularity and works for relatively small files of similar sizes.

Roussev, *et al.* [18] proposed a similarity scheme that uses partial knowledge of the internal object structure and Bloom filters. Subsequently, they developed a Rabin-style multi-resolution scheme [19] that attempts to balance performance and accuracy by maintaining hash values at several resolutions. This approach provides similarity comparisons that are flexible and meaningful, but it requires a basic understanding of the syntactic structure of the objects, which affects its generality.

Outside the realm of digital forensics, Pucha, *et al.* [12] proposed an interesting scheme for identifying similar files in a peer-to-peer network. Their scheme focuses on large-scale similarity (e.g., the same movie in different languages) and strives to select the minimum number of features necessary for identification.

2.3 Evaluation of Fingerprinting Approaches

Rabin's randomized model of fingerprinting works well on average, but suffers from problems related to coverage and false positive rates. Both these problems can be traced to the fact that the underlying data can have significant variations in information content. As a result, feature size/distribution can vary widely, which makes the fingerprint coverage highly skewed. Similarly, low-entropy features produce abnormally high false positive rates that render the fingerprint an unreliable basis for comparison.

Research in the area of payload attribution has produced more sophisticated versions of Rabin fingerprinting that seek to increase coverage (see, e.g., [5, 11, 21, 22]). These techniques manage the feature selection process so that big gaps or clusters are avoided. However, they do not consider false positives due to weak (non-identifying) features. It is important to recognize that coverage and false positives are fundamentally connected; selecting weak features to improve coverage directly increases the risk of false positive results.

3. Non-Rabin Fingerprinting

The general idea behind any similarity scheme is to select multiple characteristic (invariant) features from the data object and compare them with features selected from other objects. The collection of features can be viewed as a digital fingerprint or signature. A feature can be defined at multiple levels of abstraction, where the higher levels require more specialized processing. For the purposes of our work, we define a feature very simply as a bit sequence. In other words, we view binary data as a syntactic entity and make no attempt to parse or interpret it. This approach has obvious limitations, but is motivated by the need to develop generic, high-throughput methods that can rapidly filter large amounts of data. The expectation is that the approach would be complemented in the later stages by higher-order analysis of the filtered subset.

Our work has three main contributions. First, it presents a new feature selection scheme that selects statistically-improbable features as opposed to the randomized approach of Rabin schemes; this provides more reliable identification of characteristic features and offers even more coverage. Second, it incorporates a new approach that allows for the generic screening of inherently weak (non-identifying) features based on entropy measures; as our evaluation shows, this leads to a significant reduction in false positives. Third, it defines a new, scalable measure of similarity based on the statistical properties of Bloom filters; the measure supports the efficient comparison of objects of arbitrary sizes.

3.1 Selecting Statistically-Improbable Features

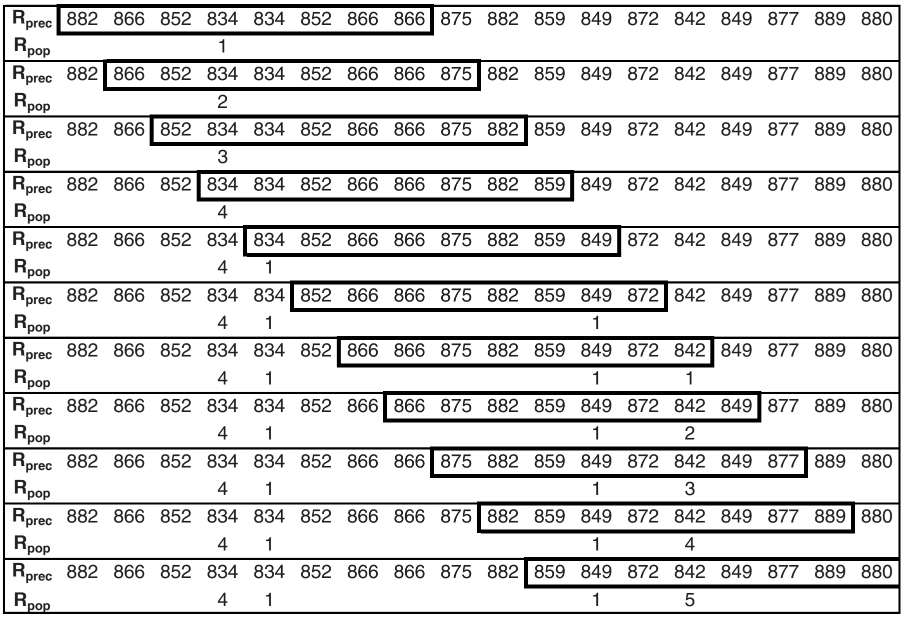
The statistically-improbable feature selection process is somewhat similar to Amazon's use of statistically-improbable phrases to characterize publications. The goal is to pick object features that are least likely to occur in other data objects by chance. The challenge is that this approach has to work for binary data (not just text) and, therefore, it is not possible to parse or interpret the data.

In this work, we consider features of size $B = 64$ bytes, which we have found to be a suitable granularity for identifying objects in disk blocks and network packets. However, there are no conceptual or implementation differences in using a different feature size. Note that a fundamental trade-off exists: the smaller the features, the higher granularity, the larger the digests and the more processing that is involved.

In all cases, the feature selection process involves the following steps:

- **Initialization:** The entropy score H_{norm}, precedence rank R_{prec} and popularity score R_{pop} are initialized to zero.

- **H_{norm} Calculation:** The Shannon entropy is first computed for every feature (B-byte sequence): $H = -\sum_{i=0}^{255} P(X_i) \log P(X_i)$, where $P(X_i)$ is the empirical probability of encountering ASCII



R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				1														
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				2														
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				3														
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4														
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4	1													
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4	1							1						
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4	1							1		1				
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4	1							1		2				
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4	1							1		3				
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4	1							1		4				
R_{prec}	882	866	852	834	834	852	866	866	875	882	859	849	872	842	849	877	889	880
R_{pop}				4	1							1		5				

Figure 1. Example R_{pop} calculation.

code i. Then, the entropy score is computed as $H_{norm} = \lfloor 1000 \times H / \log_2 B \rfloor$.

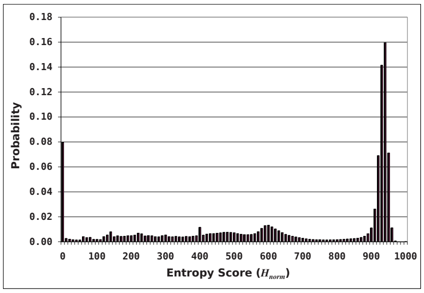
- **R_{prec} Calculation:** The precedence rank R_{prec} value is obtained by mapping the entropy score H_{norm} based on empirical observations.

- **R_{pop} Calculation:** For every sliding window of W consecutive features, the leftmost feature with the lowest precedence rank R_{prec} is identified. The popularity score R_{pop} of the identified feature is incremented by one.

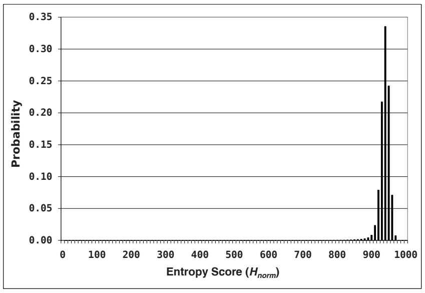
- **Feature Selection:** Features with popularity rank $R_{pop} >= t$, where t is a threshold parameter, are selected.

Figure 1 illustrates the R_{pop} calculation and feature selection steps. A snippet of 18 R_{prec} numbers from an actual computation is used; a window $W = 8$ is used for the R_{pop} calculation. Assuming a threshold $t = 4$ and feature size $B = 64$, two features are selected to represent an 82-byte piece of data.

The principal observation is that the vast majority of the popularity scores are zero or one; this is a very typical result. For an intuitive



(a) doc set.



(b) gz set.

Figure 2. Example entropy score distributions.

explanation, consider the entropy score histogram of the gz compressed data set in Figure 2(b). As expected, the overall entropy is close to the maximum, but the entropy of individual features is, in fact, normally distributed. This means that, although the entropy measures of the neighboring features are highly correlated, every once in a while, an entropy score that is less frequent is encountered (by chance). By extension, the feature itself is likely to occur less frequently. Based on our statistics, this manifests itself as a local minimum in the precedence rank, which ultimately results in a higher popularity score.

The same logic applies to other types of data; a more detailed account of our empirical observations is presented in [15]. In general, the feature selection procedure described above works on any type of data for which a reasonable (not necessarily perfect) approximation of the feature entropy distribution is available.

3.2 Filtering Weak Features

Much of the impetus to reduce the number of weak fingerprint features comes from observations of the feature entropy distribution in doc data (Figure 2(a)). As can be seen, full 8% of the data has zero entropy due to large blocks of repeated characters (mostly zeroes). Such a feature yields a raw false positive rate approaching 100%, meaning that the probability that the feature will not be unique to a specific data object is almost 100%.

This problem is by no means constrained to doc data or to zero-entropy features. Text data exhibits similar properties with raw false positive rates staying above 10% for entropy scores up to 180 [15]. At the same time, the weak features account for less than 2% of the total number of features. Eliminating weak features from consideration can reduce false positive rates with minimal effect on coverage and likely no impact on real-world applications.

In developing our `sdhash` tool, we used a 600 MB sample set of mixed, randomly-obtained files to derive the entropy score distribution and the entropy-to-precedence mapping table. During the filtering process, all features with entropy scores of 100 or below, and those exceeding 990 were unconditionally dropped from consideration. The latter decision is based on the observation that features with near-maximum entropy tend to be tables whose content is common across many files. For example, Huffman and quantization tables in JPEG headers can have very high entropy but are poor characteristic features.

3.3 Generating Similarity Digests

After the object features have been selected and filtered, the next step is to build the fingerprint representation. For this purpose, we employ a sequence of Bloom filters as in our earlier multi-resolution similarity hashing scheme [19]. A Bloom filter [1, 4] is a bit vector used for space-efficient set representation. The price paid for the space savings is the possibility that membership queries may return false positive results. To insert an element, it is hashed with k different hash functions; the results are treated as addresses inside the filter and the corresponding bits are set to one. Membership queries are handled in a similar manner; however, instead of setting the bits, they are tested to see if they are set – if all k bits are set to one, then the answer is "yes," otherwise it is "no." It is possible that the combination of bits checked during a query is set by chance, which results in a false positive. The probability of false positives can be quantified and controlled by limiting the number of elements inserted into the filter.

In our implementation, selected features are hashed using SHA-1 and the result is split into five sub-hashes, which are used as independent hash functions to insert the feature into the filter. The implementation uses 256-byte filters with a maximum of 128 elements per filter. After a filter reaches capacity, a new filter is created and the process is repeated until the entire object is represented.

One subtle detail is that before a feature is inserted, the filter is queried for the feature; if the feature is already present, the count for the number of elements inserted is not increased. This mechanism prevents the same feature from being counted multiple times, which reduces the false positive rate by forcing the inclusion of non-duplicate features; the accuracy of the similarity estimate is also increased.

3.4 Comparing Digests

The basic building block for digest comparison is the comparison of two Bloom filters. In general, the overlap between two compatible filters is a measure of the overlap between the sets they represent – the number of bits in common grows linearly with the amount of set overlap.

Consider two Bloom filters f_1 and f_2 of size m bits containing n_1 and n_2 elements ($n_1 \leq n_2$), respectively, and n_{12} elements in common. Let k be the number of hash functions used and e_1, e_2 and e_{12} be the number of bits set to one in f_1, f_2 and $f_1 \cap f_2$, respectively. Using classical Bloom filter analysis [4], the estimate of the number of expected common bits set to one is:

$$E_{12} = m\left(1 - p^{ks_1} - p^{ks_2} + p^{k(s_1+s_2-s_{12})}\right),\ p = 1 - 1/m$$

Furthermore, the estimates of the maximum and minimum number of possible overlapping bits due to chance are:

$$E_{max} = \min(n_1, n_2);\quad E_{min} = m\left(1 - p^{ks_1} - p^{ks_2} + p^{k(s_1+s_2)}\right)$$

Next, we define a cutoff point C below which all bit matches are assumed to be due to chance:

$$C = \alpha(E_{max} - E_{min}) + E_{min}$$

Thus, the similarity filter score SF_{score} of the two filters is defined as:

$$SF_{score}(f_1, f_2) = \begin{cases} -1 & \text{if } n_1 < N_{min} \\ 0 & \text{if } e_{12} \leq C \\ \left\lceil 100\frac{e_{12}-C}{E_{max}-C}\right\rceil & \text{otherwise} \end{cases}$$

where N_{min} is the minimum number of elements required to compute a meaningful score. Our implementation uses an experimentally-derived value of $N_{min} = 6$.

Given two digests SD_1 and SD_2 consisting of Bloom filters $f_1^1, \ldots, f_s^1$ and $f_1^2, \ldots, f_t^2$, respectively ($s \leq t$), the similarity digest score is formally defined as:

$$SD_{score}(SD_1, SD_2) = \frac{1}{s}\sum_{i=1}^{s} \max_{1 \leq j \leq t} SF_{score}(f_i^1, f_j^2)$$

Informally, the first filter from the shorter digest (SD_1) is compared with every filter in the second digest (SD_2) and the maximum similarity score is selected. This procedure is repeated for the remaining filters in SD_1 and the results are averaged to produce a single composite score.

The rationale behind this calculation that a constituent Bloom filter represents all the features in a continuous chunk of the original data. Thus, by comparing two filters, chunks of the source data are compared implicitly. (In fact, it is possible to store (at a modest cost) the exact range that each of the filters covers in order to facilitate follow-up work.) Thus, the size of the filters becomes a critical design decision – larger filters speed up comparisons while smaller filters provide more specificity.

The interpretation of the scores is discussed in Section 5. At this point, however, we note that the parameters, including $\alpha = 0.3$, have been calibrated experimentally so that the comparison of the fingerprints of unrelated random data consistently yields a score of zero.

4. Implementation

We have implemented the fingerprinting method in the `sdhash` tool, which is available at [17]. The usage format is `sdhash [options] {<source_file(s)> | <digest_files>}`. Users may pick one digest generation/storage option and one digest comparison option as follows:

- **Option `-g`:** This option treats the files in the list as original (source) data. For every file name `pathname/file.ext`, a corresponding `pathname/file.ext.sdbf` file containing the SD fingerprint is generated. No fingerprint comparisons are performed.
- **Option `-c`:** This option treats the files in the list as digest data and comparisons are performed.
- **Option `-f`:** This option is the combination of options `-g` and `-c`. The digest files are generated and compared.
- **Option `-m`:** This default option is the same as the `-f` option except that no digest files are created as a side effect.
- **Option `-p`:** This option causes the header(s) of fingerprint file(s) to be printed. The following example illustrates its use.

```
> ./sdhash -p 100M.doc.sdbf
100M.doc.sdbf:  bf_count: 10858, bf_size: 256,
hash_count: 5, mask:  7ff, max: 128, count: 77
```

In the example, the fingerprint consists of a sequence of 10,858 256-byte Bloom filters. Five (32-bit) sub-hashes are generated from the base SHA-1 hash and based on the bit mask, the 11 least-significant bits are used to address bits within each filter. Each filter encodes 128 features, except for the last filter, which has 77 features. The total number of features is 10,857 × 128 + 77 = 1,389,773 features.

- **Option `-2`:** This default option specifies that for n source files or digests, a total of $n-1$ pairwise comparisons should be performed: $< \#1, \#2 >, < \#1, \#3 >, \ldots, < \#1, \#n >$.
- **Option `-n`:** This option specifies that for n source files or digests, all unique pairs must be compared: $< \#1, \#2 >, < \#1, \#3 >, \ldots, < \#2, \#3 >, < \#2, \#n >, \ldots, < \#2, \#n >, \ldots, < \#n-1, \#n >$.

The `sdhash` output consists of three columns. The first two columns list the files that are compared; the third column gives the corresponding SD scores.

5. Cross-Set Fragment Detection Experiments

This section discusses a "needle in a haystack" fragment detection scenario. Given a relatively small snippet of data such as a disk block or network packet ("needle"), the goal is to determine whether or not parts of it are present in the large set of data ("haystack").

Based on the scenario, the `sdhash` parameters were tuned to work for fragments in the 1024-byte to 4096-byte range. We also studied the boundary case of a 512-byte fragment to understand the behavior outside the design zone. As the results show, the method degrades gracefully; however, the accuracy of the results inherently drops because it becomes more difficult to find 64-byte characteristic features. For reasons of space, the scenario involving the detection of similar objects is not presented in this paper; it will be the subject of a future article.

5.1 Experimental Setup

This section presents the data, parameters and metrics used in the experimental evaluation.

Data Six sample 100 MB document sets from the NPS Corpus [6] were used in the experiments: (i) Microsoft Word documents (`doc`); (ii) HTML documents (`html`); (iii) JPEG images (`jpg`); (iv) Adobe PDF documents (`pdf`); (v) Plain text documents (`txt`); and (vi) Microsoft Excel spreadsheets (`xls`). In addition, a seventh 100 MB pseudorandom data set (`rnd`) was obtained from `/dev/urandom`. The `rnd` set represents a calibration benchmark because its content is unique and no features selected from it appear in the other sets. It is infeasible to provide similar guarantees for the remaining sets. The results obtained for the `rnd` set essentially correspond to a best-case scenario for what can be achieved in practice. Therefore, it is used as a benchmark to evaluate how close the other results come to the best case.

Parameters Based on the primary scenario and our preliminary evaluation of SD hashes, the following parameters were chosen for the fingerprints:

- **Feature Selection:** The feature size $B = 64$ bytes, i.e., all possible sliding 64-byte sequences in an object were considered. The window size $W = 64$ bytes, i.e., one representative feature was selected from every 64 consecutive features based on the entropy measures defined earlier. The threshold $t = 16$, i.e., only the representative features selected in at least 16 consecutive windows were considered.

- **Bloom Filters:** Upon selection, each feature was hashed using SHA-1 and the resulting 160 bits of the hash were split into five sub-hashes of 32 bits, each of them treated as an independent hash function. The actual similarity digest is a sequence of 256-byte Bloom filters with 128 elements (features) per filter. Thus, the expected false positive rate of the Bloom filter is 0.0014 for an individual membership query.

- **Fragment Size:** Four different fragment sizes of 512, 1024, 2048 and 4096 bytes were used to evaluate the behavior of the similarity measure in the range of the fragment sizes of interest.

Evaluation Metrics We considered three basic measurements: detection rates, non-classification rates and misclassification rates. The first step was to generate sample fragment sets for every combination of fragment size and data set. These were obtained by picking random file-offset combinations and extracting a fragment of the appropriate size. This gave rise to 28 ($= 4 \times 7$) fragment sets, each with 10,000 samples. SD fingerprints were generated for each reference set and sample, which were then compared depending on the scenario:

- **Detection Rate:** This metric assesses the likelihood that the SD hash method correctly attributes a sample set to its source. In other words, the fragment fingerprints are compared to the source set fingerprint for every sample set. Note that the classification results depend on the choice of minimum score threshold value – the higher the value, the lower the detection rate.

- **Non-Classification Rate:** This metric assesses the likelihood that the SD hash method rejects a fragment set as not containing enough characteristic features for reliable classification. This is the equivalent an "I don't know" answer to a similarity query. Note that this rate is solely a function of the fragment content and does not depend on the comparison target; thus, the sample can be identified as being weak before comparisons are performed. Intuitively, the non-classification rate can be expected to be very close to zero for high-entropy (random, compressed or encrypted) data. Furthermore, an elevated non-classification rate is an implicit indicator that the comparison results are less reliable due to small fragment size and/or low sample entropy.

- **Misclassification Rate:** Recall that an SD hash comparison produces a score in the 0 to 100 range (-1 for non-classification). Given a random sample that is present in the reference file, it is not guaranteed that the returned result will be 100. In particular, it is quite likely (depending on alignment) that the features selected from the sample will spread into more than one Bloom filter in the fingerprint of the source.

  In practice, this implies that the selected threshold should balance the probabilities of false negatives and false positives. To cumulatively capture these errors, the false negative and false positive rates are summed to produce the misclassification rate. Evidently, the misclassification rate is a function of the threshold value chosen to separate false negatives and false positives. The closer the threshold is to zero, the higher the probability for false positives; conversely, the closer the threshold is to 100, the higher the probability of false negatives.

  Ideally, there is a range of SD scores for which the misclassification rate is zero, so a safe threshold could be picked in the range. In some cases, such perfect ranges do, in fact, exist. Most of the time, however, the goal is to minimize the misclassification rate. To obtain the misclassification rate estimates, the scores of 10,000 true samples from a source set were compared with those of 10,000 samples taken from each of the other sets. Then, the misclassification rate was identified for every possible value of the threshold (1-99); the best choice showed up as a global minimum in the plot.

5.2 Experimental Results

Space constraints prevent us from presenting the complete set of results (these will be published separately as a technical report). Fortu-

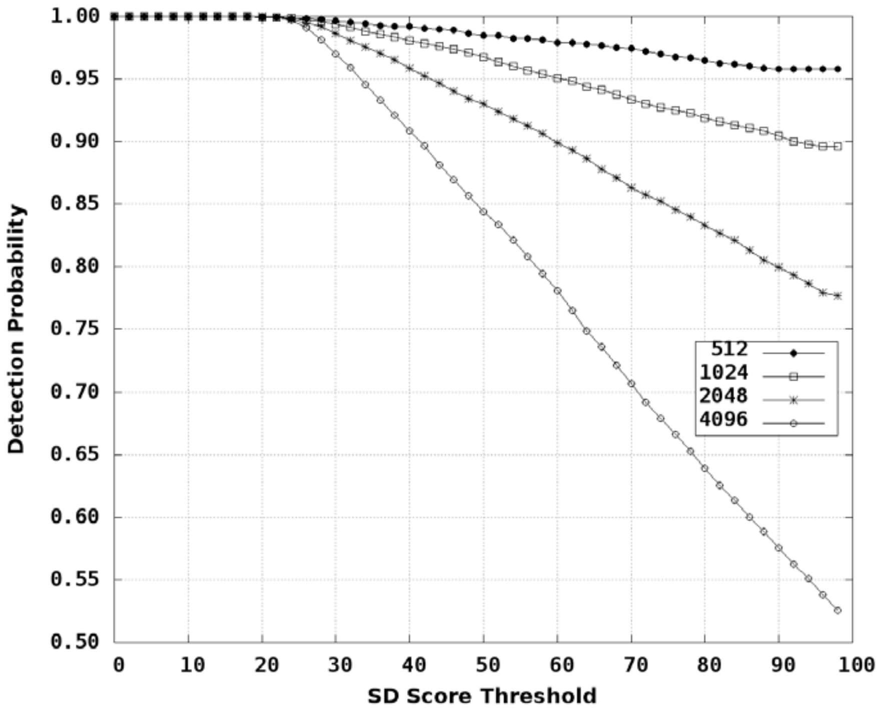


Figure 3. Detection rates for the `txt` reference set.

nately, the observed behavior was consistent for all the data sets. As a result, we only present the detailed results for the `txt` set.

Detection Rates Figure 3 presents the the `txt` detection performance (y-axis) as a function of the SD score threshold (x-axis) and the fragment size (512, 1024, 2048 and 4096 bytes). The first observation is that a threshold score of up to 22 yields near-perfect (0.999+) detection rates for all fragment sizes. Detection rates drop approximately linearly beyond this value, with rates for larger fragments dropping faster than those for smaller fragments. The latter result is expected because, as the fragment size grows, so does the probability that the fragment features selected will end up in multiple Bloom filters in the digest of the original file. This is best illustrated by the rightmost points of the curves, which represent the fraction of true positives that generate a score of 100.

Non-Classification Rates Table 1 shows the non-classification rates for various test sets and fragment sizes. The decision to refuse classification is based on the requirement that a fragment must contain a

Table 1. Non-classification rates for various test sets and fragment sizes.

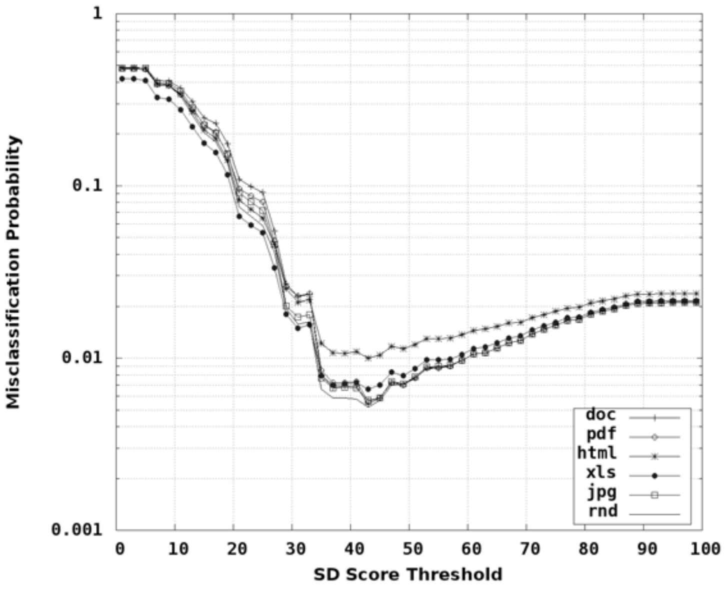
Test	**Fragment Size**			
Set	**512**	**1024**	**2048**	**4096**
`doc`	0.2383	0.1291	0.0764	0.0435
`html`	0.0995	0.0059	0.0025	0.0008
`jpg`	0.0281	0.0089	0.0045	0.0033
`pdf`	0.0533	0.0198	0.0163	0.0157
`rnd`	0.0166	0.0000	0.0000	0.0000
`txt`	0.0860	0.0192	0.0060	0.0031
`xls`	0.0706	0.0113	0.0058	0.0024

minimum of six unique selected features for the comparison to proceed. The cutoff point was obtained empirically based on the observation that misclassification rates escalate rapidly below this threshold without enhancing detection. The non-classification rates for 512-byte fragments are significantly higher than those for larger fragments. Note that the non-classification rate is zero for 1024-, 2048- and 4096-byte `rnd` fragments.

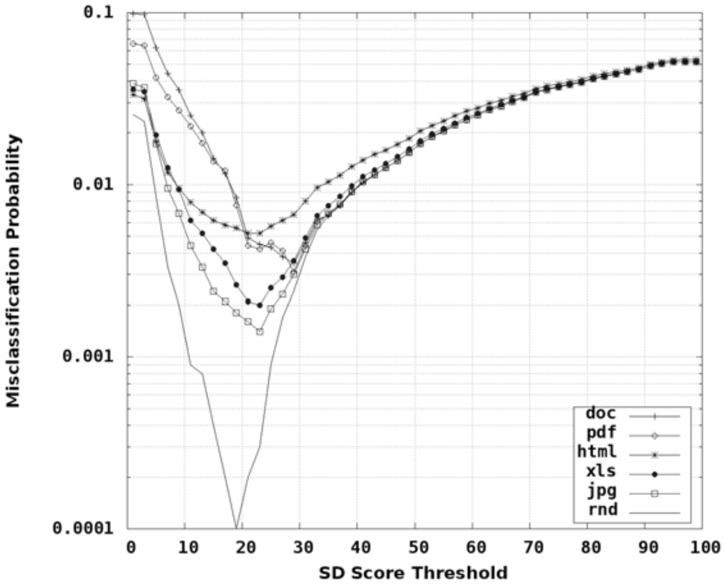
An important result is that the `doc` set exhibits high non-classification rates for all four fragment sets. This is not entirely surprising given our earlier survey of feature entropy distributions for different sets. Specifically, we showed that 8% of all potential features in the `doc` set have zero entropy (Figure 2(a)). The actual problem areas expand beyond this to encompass neighboring areas where entropy is too low.

It is important to emphasize that our design philosophy is to reject weakly-identifiable data and not classify them. This decision is justified because reliable estimates of error rates cannot be obtained without excluding weak data.

Misclassification Rates Figure 4 presents the misclassification results for all 512- and 1024-byte fragments with respect to the `txt` reference set as a function of the chosen SD score threshold values. Since the y-axis uses a logarithmic scale, the zero values are replaced with 0.0001 to enable visualization. Figure 4(a) is representative of the behavior observed across all experiments with 512-byte fragments. On the other hand, Figure 4(b) is representative of all the experiments involving 1024-, 2048- and 4096-byte fragments (larger fragments produce marginally better results). This is welcome news because we seek classification threshold values that work consistently across all sets. The observed consistency demonstrates that the method is stable and works well across the spectrum of data.



(a) 512-byte fragments.



(b) 1024-byte fragments.

Figure 4. Misclassification rates for the `txt` reference set.

Table 2. Misclassification rates for various test sets and fragment sizes.

Test	**Fragment Size**							
Set	**512**		**1024**		**2048**		**4096**	
	min	**max**	**min**	**max**	**min**	**max**	**min**	**max**
`rnd`	0.0050	0.0100	0.0006	0.0050	.0003	.0040	.0003	0.0040
`doc`	0.0050	0.0100	0.0002	0.0050	.0003	.0050	.0002	0.0040
`html`	0.0060	0.0100	0.0003	0.0050	.0005	.0055	.0000	0.0035
`jpg`	0.0050	0.0120	0.0005	0.0055	.0002	.0050	.0003	0.0040
`pdf`	0.0060	0.0120	0.0002	0.0050	.0002	.0050	.0002	0.0040
`txt`	0.0050	0.0100	0.0002	0.0055	.0002	.0050	.0000	0.0035
`xls`	0.0060	0.0130	0.0002	0.0055	.0002	.0050	.0150	0.0180

Figure 4(a) demonstrates that there is room for varying the optimization criterion, and that a case can be made for a number of possible threshold values in the 37-49 range. Upon inspecting all the graphs, 43 emerges as the best candidate because it consistently achieves near-optimal results across all the 512-byte fragment experiments. A value of 21 yields the most consistent results for the 1024-, 2048- and 4096-byte fragments.

Table 2 summarizes the results for all the test sets using the chosen threshold values. Each cell provides the minimum or maximum misclassification rate observed for a fragment of a particular size. Note that the top row of the table (`rnd`) is the reference best-case scenario, and the other six sets produce very similar results.

Storage and Throughput On the average, storing a similarity digest along with the chosen parameters requires about 2.6% of the original source data. However, it is possible to shrink the on-disk representation down to 2.4% using standard zip compression. This strikes a good balance between accuracy and compactness of representation – a commodity server costing $5,000 can be equipped with 32 to 48 GB RAM, which would support the in-memory representation of 1.25 to 1.75 TB of data.

The current implementation is capable of generating SD hashes at the rate of approximately 30 MB/sec/core on a modern processor. Thus, the SD fingerprinting method can be applied during the imaging process and would be able to identify artifacts during target acquisition.

6. Conclusions

Our method for generating data fingerprints based on statistically-improbable features engages a generic entropy-based scheme to efficiently select features from binary data. Prior work relies on randomized feature selection, which provides uneven coverage and produces relatively high false positives for low-entropy data. The method enables features with low information content to be filtered, thereby reducing false positives.

Experimental evaluations of the performance of the `sdhash` implementation on small (1 to 4 KB) data fragments from six common file types demonstrate the robustness of the method. The error rate as represented by misclassified fragments (including false positives and false negatives) does not exceed 0.0055, implying a correct classification rate of 0.9945. The success of the technique is due to the fact that the algorithm flags fragments that do not contain enough identifying features. The space requirements for the generated similarity digests do not exceed 2.6% of the source data, which makes it possible to maintain digests of images up to 1.5 TB in memory.

Our future research will refine and optimize the tool, and perform tests on large forensic images using digests generated from the NSRL set. We also plan to explore the capabilities of the tool by tuning its parameters for smaller and larger features. Along the same lines, we plan to add a preprocessing step that will recognize common header features that tend to produce false positives.

References

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

[2] S. Brin, J. Davis and H. Garcia-Molina, Copy detection mechanisms for digital documents, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 398–409, 1995.

[3] A. Broder, S. Glassman, M. Manasse and G. Zweig, Syntactic clustering of the web, *Computer Networks and ISDN Systems*, vol. 29(8-13), pp. 1157–1166, 1997.

[4] A. Broder and M. Mitzenmacher, Network applications of Bloom filters: A survey, *Internet Mathematics*, vol. 1(4), pp. 485–509, 2005.

[5] C. Cho, S. Lee, C. Tan and Y. Tan, Network forensics on packet fingerprints, *Proceedings of the Twenty-First IFIP Information Security Conference*, pp. 401–412, 2006.

[6] Digital Corpora, NPS Corpus (digitalcorpora.org/corpora/disk-images).

[7] J. Kornblum, Identifying almost identical files using context triggered piecewise hashing, *Digital Investigation*, vol. 3(S1), pp. S91–S97, 2006.

[8] U. Manber, Finding similar files in a large file system, *Proceedings of the USENIX Winter Technical Conference*, pp. 1–10, 1994.

[9] M. Mitzenmacher, Compressed Bloom filters, *IEEE/ACM Transactions on Networks*, vol. 10(5), pp. 604–612, 2002.

[10] National Institute of Standards and Technology, National Software Reference Library, Gaithersburg, Maryland (www.nsrl.nist.gov).

[11] M. Ponec, P. Giura, H. Bronnimann and J. Wein, Highly efficient techniques for network forensics, *Proceedings of the Fourteenth ACM Conference on Computer and Communications Security*, pp. 150–160, 2007.

[12] H. Pucha, D. Andersen and M. Kaminsky, Exploiting similarity for multi-source downloads using file handprints, *Proceedings of the Fourth USENIX Symposium on Networked Systems Design and Implementation*, pp. 15–28, 2007.

[13] M. Rabin, Fingerprinting by Random Polynomials, Technical Report TR1581, Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts, 1981.

[14] S. Rhea, K. Liang and E. Brewer, Value-based web caching, *Proceedings of the Twelfth International World Wide Web Conference*, pp. 619–628, 2003.

[15] V. Roussev, Building a better similarity trap with statistically improbable features, *Proceedings of the Forty-Second Hawaii International Conference on System Sciences*, pp. 1–10, 2009.

[16] V. Roussev, Hashing and data fingerprinting in digital forensics, *IEEE Security and Privacy*, vol. 7(2), pp. 49–55, 2009.

[17] V. Roussev, `sdhash`, New Orleans, Louisiana (roussev.net/sdhash).

[18] V. Roussev, Y. Chen, T. Bourg and G. Richard, `md5bloom`: Forensic filesystem hashing revisited, *Digital Investigation*, vol. 3(S), pp. S82–S90, 2006.

[19] V. Roussev, G. Richard and L. Marziale, Multi-resolution similarity hashing, *Digital Investigation*, vol. 4(S), pp. S105–S113, 2007.

[20] V. Roussev, G. Richard and L. Marziale, Class-aware similarity hashing for data classification, in *Research Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 101–113, 2008.

[21] S. Schleimer, D. Wilkerson and A. Aiken, Winnowing: Local algorithms for document fingerprinting, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 76–85, 2003.

[22] K. Shanmugasundaram, H. Bronnimann and N. Memon, Payload attribution via hierarchical Bloom filters, *Proceedings of the Eleventh ACM Conference on Computer and Communications Security*, pp. 31–41, 2004.

Chapter 16

REFINING EVIDENCE CONTAINERS FOR PROVENANCE AND ACCURATE DATA REPRESENTATION

Bradley Schatz and Michael Cohen

Abstract It is well acknowledged that there is a pressing need for a general solution to the problem of storing digital evidence, both in terms of copied bitstream images and general information that describes the images and context surrounding a case. In a prior paper, we introduced the AFF4 evidence container format, focusing on the description of an efficient, layered bitstream storage architecture, a general approach to representing arbitrary information, and a compositional approach to managing and sharing evidence. This paper describes refinements to the representation schemes embodied in AFF4 that address the accurate representation of discontiguous data and the description of the provenance of data and information.

Keywords: Evidence containers, representation, provenance, tool interoperability

1. Introduction

One of the principal challenges in digital forensics is to deal with the rapidly growing volume and complexity of information that is the subject of investigations [4]. The acquisition and analysis of digital evidence are hampered by the lack of interoperability between forensic analysis tools; important forensic information is often unused by analysis tools because it is locked within proprietary file formats or free text. Access to case data is hampered by closed abstraction layers and the inefficiencies imposed by the need to manually copy data in order to process it with task-specific tools. Finally, the management of evidence is slowed by format conversion and storage bandwidth limitations.

Digital forensic practitioners have largely settled on the raw (`dd`) and Expert Witness Format (EWF) evidence storage formats for hard drive images. This has enhanced the ability – in the storage forensics field at least – to acquire and analyze evidence using a variety of tools of commercial and open source lineage.

Such formats are, however, poor surrogates for the original evidence. A raw image fails to distinguish between sectors containing bytes with the value zero and those where a read error has occurred. A raw image does not record provenance-related information pertaining to the drive such as the drive geometry and drive configuration overlays, nor does it record the activities performed on the drive or image. The EWF format popularized by the EnCase tool is similarly limited; however, it does overcome some of the weaknesses of a raw image by recording limited meta information related to the image within the container itself.

In practice, provenance information describing the evidence and its outside context rarely becomes grist for the automated forensic mill, mainly because it is collected and recorded manually by investigators in a variety of formats, including handwritten free text and *ad hoc* file formats. New approaches are required to represent digital evidence, both in terms of raw bits and bytes on storage media (data) and information describing related artifacts, entities and analytic results.

In an earlier paper [9], we introduced the AFF4 evidence container format, which is designed to store arbitrary evidence images, context-related information and analysis results within a unified container format. This paper describes refinements to AFF4 that address the accurate representation of discontiguous data and describe the provenance of data and information.

2. Ideal Evidence Container

The ideal evidence container would present to the investigator a perfect surrogate of the original physical evidence, whether it is a hard drive, mobile phone flash memory or computer RAM. Such a container would fully describe the characteristics, behavior, content and context of the original evidence that it represents. These include:

- **Data Content:** Multiple streams (HPA, DCO); hierarchical data relationships (logical imaging); addressing windows (RAM holes, bad sectors); addressing schemes (block size, CHS/LBA); SMART status.

- **Physical Characteristics:** Make; model; serial number; interface (SATA, SCSI, etc.).

- **Context:** Environment in which the hard drive existed; case-related information.

- **Behavior:** Error codes related to bad sectors.

A perfect fidelity surrogate – even if it were technically feasible – would not be entirely desirable. Such a "virtual hard disk" would quickly frustrate the investigator by reliably imposing characteristics such as read retries on bad sectors, I/O bandwidth limitations and seek latency. Accordingly, the ideal container would sacrifice fidelity to satisfy orthogonal operational concerns such as:

- **Efficiency:** Storage space minimization; random access performance; I/O speed.
- **Authentication:** Cryptographic signing; hash storage.
- **Privacy:** Encryption; redaction.
- **Resilience:** Tolerance to underlying storage media failures.

3. Current State of Evidence Containers

From the original raw evidence format, evidence containers evolved to incorporate seekable compression [9], embedded authentication and integrity mechanisms such as hashes and CRC, and storage of a small number of fields for describing images. More recently, the demand for logical imaging has resulted in the emergence of a proprietary evidence container format that supports the storage of multiple streams of data along with file-oriented metadata (EnCase Logical Evidence File).

This current breed of commercial evidence containers does not address the characteristics of evidence sources. For example, hard drives may contain multiple address spaces, depending on whether features such as host protected areas or drive configuration overlay are enabled. Images of computer RAM require the consideration of holes in which no data exists. Furthermore, the evidence containers fail to address the storage of general information that is of relevance.

The research community has proposed a number of container formats. The Advanced Forensics Format (AFF) [10, 11] introduced the storage of arbitrary metadata within an evidence container, privacy via encryption and redaction, and resilience via fault tolerance. Digital evidence bags [15] store arbitrary textual information along with images in the same container. Sealed digital evidence bags [14] employ a composition framework for evidence containers based on a linked information model.

We recently introduced the AFF4 evidence container format [9], which defines an efficient, seekable, compressed storage format for multiple data object images, a novel and powerful data model that enables the composition of data objects from other data objects, and an information

Table 1. Comparison of representational capabilities of container formats.

	Data Representation							
	raw	**EWF**	**sgzip**	**LE1**	**DEB**	**SDEB**	**AFF1**	**AFF4**
Single Image Storage	Y	Y	Y	N	Y	Y	Y	Y
Multiple Image Storage	N	N	N	N	Y	Y	N	Y
Hierarchical Image Storage (Logical Imaging)	N	N	N	Y	Y	Y	N	Y
Addressing Windows (Discontiguous Images)	N	N	N	N	?	N	N	Y
Data Composition	N	N	N	N	N	N	N	Y
Seekable Compression	N	Y	Y	?	N	N	Y	Y
	Information Representation							
	raw	**EWF**	**sgzip**	**LE1**	**DEB**	**SDEB**	**AFF1**	**AFF4**
Metadata Storage	N	Y	N	Y	Y	Y	Y	Y
Arbitrary Metadata Storage	N	N	N	?	Y	Y	Y	Y
Arbitrary Information Storage	N	N	N	N	Y	Y	N	Y
Formal Information Model	N	N	N	N	N	Y	N	Y
Composable Information Model	N	N	N	N	N	Y	N	Y

representation approach based on a linked information model. Table 1 compares the characteristics of AFF4 and those of other evidence container formats.

4. AFF4 Data and Information Models

The general design goals of AFF4 are to provide an open and extensible evidence container that facilitates the storage, composition and sharing of arbitrary types of digital evidence, information and analysis results. AFF4 defines two interrelated models, one for representing and documenting information, and the other for storing, referring to and transforming bitstream data. The two models are linked by a naming scheme in which items of relevance are identified using globally-unique identifiers.

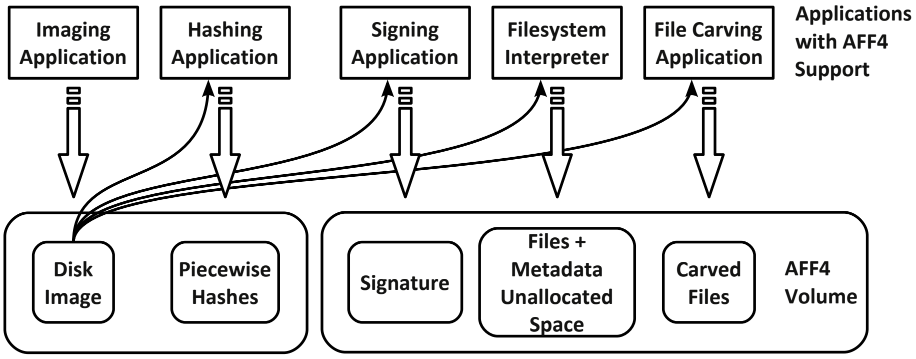


Figure 1. Layered application of forensic tools with the AFF4 container.

The AFF4 data model is specifically designed to facilitate the storing, sharing and referencing of data without imposing the storage bandwidth burden of copying data, while providing efficiency in terms of storage space, random access latency and I/O bandwidth. Likewise, the AFF4 information model is designed to facilitate the description of forensically-relevant information, including evidence, case context and general analytic results.

The "by reference" approach of the AFF4 architecture facilitates the successive layered application of discrete task-specific tools on the analytic results and data abstractions of tools operating at lower layers of abstraction. In the architecture, the analysis results of a tool are persistent when using the AFF4 information and data models in a new AFF4 container or in an existing container. This significantly differs from other approaches that either keep analysis results as intermediary structures in the working memory of a tool (as in monolithic applications like EnCase) or dump the results to an *ad hoc* (and likely non-machine-readable) document.

Figure 1 presents the flow of information and data between AFF4-aware forensic applications and AFF4 evidence containers. In this example, an examiner employs an AFF4 imaging application to create two disk images in a single container. Each image has two virtual address spaces overlaying it: one for the regular portion of the disk and the other that includes the HPA portion of the disk. The examiner uses the GUI of the tool to enter case-relevant and context-related information associated with the images; meanwhile, the imaging tool records provenance-related information obtained directly from the drives. Finally, the tool records the MD5 hash of the entire disk and the information related to media failures.

Upon returning to the laboratory, the examiner proceeds to preserve the evidence. A piecewise hash of the disk is created using SHA-256 or some other hash algorithm (this is performed in the laboratory because a hash algorithm such as SHA-256 is too slow to use in the field). The computed hash values are stored in a new container, which refers to the original container.

The investigator then uses a public key to cryptographically sign the information associated with the image, including the piecewise hashes. This signature is stored in the new container.

Automated processing of the image may involve a filesystem interpreter to create virtual file instances for each file in the image, instances for deleted files and instances for unallocated space. These are also placed in a new evidence container for subsequent consumption by other tools. The unallocated space is then processed by a file carver, the results of which are stored as virtual file instances in a new evidence container, resulting in zero copy file carving [12].

This scenario demonstrates the feasibility of the AFF4 architecture with respect to forensic imaging, piecewise hashing, signing and using filesystem interpreter applications.

5. AFF4 Naming Scheme

The fundamental premise of representing both data and information within AFF4 is that an object instance is created for any virtual or real-world entity – be it a disk partition or a suspect. The object instance acts as a surrogate for the real or virtual entity. Surrogates are identified by associating a globally-unique identifier with each entity. We use both uniform resource locators (URLs) and uniform resource names (URNs) as entity identifiers.

We define structured URNs, which we call AFF4 URNs, for identifying surrogates. An AFF4 URN is made globally unique by including a GUID in its form:

```
urn:aff4:195bdf58-1bc9-4ba4-9a9c-f1c312673fbf
```

We use a variant of the ZIP file format for a default container. Thus, an investigator who visually examines an AFF4 volume containing a disk image would readily identify AFF4 URNs in the following locations:

- The information segment, which is a file in an AFF4 container, contains a serialization of all the objects within the volume. The base attribute of the RDF serialization refers to the volume URN.

- Image file segments are present in a folder within the ZIP file, where the folder name is a filesystem-friendly encoding of the URN that uniquely identifies the image. For example, a folder named `urn%3Aaff4%3A195bdf58-1bc9-4ba4-9a9c-f1c312673fbf` might refer to the image stream `urn:aff4:195bdf58-1bc9-4ba4-9a9c-f1c312673fbf/`. Archive filenames can be shortened relative to the volume URI. For example, the folder `diskimage` corresponds to the fully-qualified URI `urn:aff4:volume_URI/diskimage`.

- Information segments (described in the next section) contain URNs within their text.

URLs, which are typically used to uniquely identify a piece of terminology, may be found as text within information segments and maps. For example, we use the following URL to represent the concept of an "Image:"

```
http://afflib.org/2009/aff4#Image
```

6. Refining the AFF4 Information Model

The original AFF4 information model was inspired by the Resource Description Framework (RDF) [17], the data model that underlies the Rich Site Summary (RSS) feeds used by blogging software. The RDF data model facilitates object-oriented modeling with the key difference that objects in the RDF universe have unique names and properties, and the attributes and relationships of individual objects may be published in different documents.

While the original information model is intuitive and simple to implement, it suffers from several shortcomings:

- The method of mapping the information model to container segments (i.e., serialization scheme) results in inefficient storage when large numbers of information instances are described.
- The information model is not expressive enough to describe provenance-related information.
- The AFF4 serialization scheme is verbose, which makes it difficult to read.
- The lack of value types leads to ambiguity in interpreting values. Also, there is no syntactic means for distinguishing between values and URI references.

For these reasons and for standardization and interoperability, we abandon the *ad hoc* RDF variant and instead adopt RDF in its entirety as the information model. Under the new scheme, information is stored in information segments whose suffixes represent the RDF encodings.

For example, the name `information.turtle` in the AFF4 container, refers to an RDF serialization using the Turtle encoding scheme [2]. For conciseness and readability, we use Turtle as the default RDF serialization syntax. The URN of the serialization component is significant, with the path component being interpreted as the graph name in which the encoded RDF exists.

7. Provenance of Information

This section describes a general approach for expressing the provenance of statements in the AFF4 universe. Provenance statements are required to express information such as which tool generated which image or analysis products, and to sign statements.

The original AFF4 provides for provenance-related statements in the particular case of signing. However, the identity object, which implements signatures, presents some difficulties in the case of provenance statements:

- The semantics of the statement file in relation to its enclosing instance is inconsistent with the information model.
- Statements within a statement file are also made elsewhere within the container.
- Verifying the signatures of statements requires that the statements be in the exact order and syntax in which they exist in the signature file.

A general solution to provenance requires a method for referring to a set of RDF statements as a whole. Such statements about statements are called "reified statements" in the knowledge representation literature. A simple example is: Dick said "The serial number of the hard drive is `ZX322o91` and its hash value is `13343af423d`."

The subject Dick is making a statement covering two separate statements: "The serial number of the hard drive is `zx322o91`" and "The hash value of the hard drive is `13343af423d`."

A widely acknowledged problem of RDF is its limited ability to express reified statements [7]. Named graphs constitute a solution to the problem of reification in RDF, with the TriG language emerging as a syntax for encoding reified information [6]. A named graph is simply a collection of RDF statements that can be identified by an unambiguous name. Using the TriG syntax, the reified statement above can be expressed as:

```
1    @prefix G1:  <urn:aff4:19857a87-a190b2f87>
2    @prefix Hdd1: <urn:aff4:652e4027-27fab2941>
```

```
3    @prefix aff4: <http://afflib.org/2009/aff4#>
4    @prefix Dick: <urn:aff4:652e4027-27fab2941>
5
6    G1: {
7      Hdd1: aff4:serialNumber "zx322o91"
8      Hdd1: aff4:hash "13343af423d"
9    }
10
11   Dick: aff4:said <G1:>
```

In the listing above, Lines 1-4 define namespace identifiers that are substituted when they occur elsewhere in the document. For example, in Line 7, `aff4:serialNumber` is interpreted to mean the URL `http://afflib.org/2009/aff4#serialNumber`. This uniquely-identified vocabulary term is defined to have the meaning of a serial number. In Lines 6-9, `G1:` is the unique identifier for the named graph that contains the two statements referred to in our example. Named graph identifiers may be referred to in the subject and object parts of RDF statements. Finally, in Line 11, there is a single statement that refers to the named graph `G1:`.

Following this approach, we refine the semantics of the AFF4 information model to imply that for any information segment, the statements implied by interpreting the content of the segment are defined to exist within a named graph based on the following conventions:

- The graph name is the URN of the volume when the information segment is in the root of the volume.
- The graph name is the URN interpretation of the path when the information segment exists in a sub-path of the volume.

Consider, for example, an AFF4 ZIP container containing a ZIP file comment `urn:aff4:6cd61-52398e-4942ea` and the two information segments in the listing:

```
/information.turtle
/urn%3Aaff4%3A19d6cd61-598e-49ff/information.turtle
```

The RDF statements contained in the first segment would be interpreted to exist in a named graph with the URN of the AFF4 volume `urn:aff4:6cd61-52398e-4942ea`. The RDF triples contained in the second segment would be interpreted as being contained in the graph named `urn:aff4:19d6cd61-598e-49ff` after decoding the filesystem-friendly encoding.

Provenance-related statements employ the named graph semantics in their statements. Consider, for example, the recording of the provenance of the information describing an image generated by the command line tool `aff4imager`. An abridged set of statements is presented in the listing:

```
1    @prefix G1: <urn:aff4:19857a87-a190b2f87>
2    @prefix G2: <urn:aff4:0a1fc78a-927bfacef>
3    @prefix T1: <urn:aff4:652e4027-ffff01199>
4    @prefix I1: <urn:aff4:9003027a-11199ffff>
5    @prefix aff4: <http://afflib.org/2009/aff4/#>
6
7    G2: {
8      T1: aff4:name "aff4imager"
9      T1: aff4:vendor <http://aff.org/>
10     T1: aff4:asserts G1:>
11     T1: aff4:type aff4:AcquisitionTool.
12     T1: aff4:version "0.2"
13     I1: aff4:type aff4:Image
14     I1: aff4:hash "3897450fa18094b13"^^aff4:md5
15   }
```

In this example, we define an instance `T1:` that represents the tool and an instance `I1:` that represents the image. The `aff4:asserts` predicate is used to specify that the tool "asserted" the information contained in the graph `G1:`.

By identifying instances of type `aff4:Tool` and then identifying the graph in which the statements are located, downstream consumers of AFF4 containers would be able to identify the tool that generated specific information and data. While it is not related to provenance, note that the type `^^aff4:md5` in Line 14 indicates the data type of the text preceding it within quotes. In this case, it indicates that the value of the `aff4:hash` predicate is the hex-encoded MD5 message digest of the image.

8. Authentication and Non-Repudiation

With a means for referring to sets of statements in place, the approach to authentication and non-repudiation of evidence can be been refined. We conceptualize the relationship of signing containers in a manner similar to the approach proposed in [6]. An identity remains a person or entity as in the earlier AFF4 implementation. A warrant graph is a set of statements that record the intentions or beliefs of an identity about another set of statements, whether it be asserting, denying or quoting. The identity vouches for the truth of the warrant graph by signing the graph with a public key.

The following listing contains a warrant graph that refers to the `aff4imager` information presented in the listing above:

```
1     @base: <aff4://1b056380-a0911-f06721>
2     @prefix G2: <urn:aff4:0a1fc78a-927bfacef>
3     @prefix G3: <aff4://19857a87-a190b-2f87ab>
4     @prefix A1: <aff4://502ffb11-00f10-7fcbaf>
5
6     G3: {
7       G2: aff4:assertedBy G3:
8       G3: aff4:hash "TljN2NiNzExMmEwM2MxNG"
                                ^^aff4:canonical-sha256
9       G3: aff4:authority A1:
10      A1: aff4:certificate A1:/cert.pem
11      G3: aff4:signature "XSAFfbgEL5C8vA1W/W-="
                                ^^aff4:canonical-sha256-rsa
12    }
```

The following observations can be made with regard to the listing:

- Line 7 refers to the graph `G2:` from the previous listing. This statement indicates that the warrant graph asserts the truth of `G2:`. Any number of named graphs may be asserted (or denied or other) within a warrant graph.

- The graph digest of graph `G3:` is stated in Line 8. For serialization independence, a graph digest is a message digest with the canonical form of a set of triples in a named graph rather than the serialized syntax. This facilitates the verification of the authenticity of the target graph. The graph canonicalization method is specified by the type parameter `aff4:canonical-sha256`, and is a variant of the graph canonicalization algorithm in [5] and the digest and signature methods defined in the XML signature standard [1]. The type parameter additionally indicates that the graph digest uses the SHA-256 digest on the canonical graph.

- Line 9 states that the identity `A1:` authorizes the warrant graph.

- Line 10 states that the public key certificate of `A1:` is found at the URN `A1:/cert.pem`.

- Line 11 states the signature of `G3:` (warrant graph). The type `^^aff4:canonical-sha256-rsa` indicates the method by which the signature was constructed, which was to take the warrant graph, canonicalize as above, take the SHA-256 hash, sign it using the RSA private key of the authority `A1:`, and then encode it using Base64.

The verification of a signed AFF4 container involves identifying a signed warrant graph, removing the `aff4:signature` statement from the graph, canonicalizing the resulting graph, and then re-verifying the calculated SHA-256 RSA signature. Recalculating the graph digests of graphs asserted by the warrant graph further authenticates the information contained in the graphs.

The use of named graphs, graph digests and graph signatures facilitates the piecewise generation of authenticable and non-repudiable information in the AFF4 universe.

9. AFF4 Data Model

AFF4 defines two abstractions for storing and representing bitstream data: the Stream and the Map. Instances of each of these abstractions are identified by an AFF4 URI.

The lowest layer of abstraction for data storage in the AFF4 data model is the Stream, an abstraction of a contiguous, randomly-accessible byte sequence. AFF4 defines a number of implementations for storing and accessing Streams ranging from an efficient randomly-accessible compressed container to a flat raw file.

The Map abstraction similarly represents a contiguous, randomly-accessible byte array; however, it is composed of byte arrays from multiple stream sources. Defining virtual data objects as comprising portions of existing concrete data sources enables references to data objects within images (e.g., files) or data objects composed of multiple images (e.g., reconstructed RAID volumes). Maps are used as the fundamental building block for representing the data portions of files, partitions, unallocated space and reconstructions of virtual RAID volumes from images. Backward compatibility with EWF images is provided by creating a virtual image that maps to each compressed EWF segment.

10. Refining the AFF4 Data Model

AFF4 applications have revealed two shortcomings of the data model: (i) referring to subranges of data within a Stream is heavyweight, requiring the definition of a Map; and (ii) the data model does not support discontinuities that occur in evidence sources such as RAM and faulty hard drives.

10.1 Referring to Byte Ranges

AFF4 requires a means to refer to arbitrary address ranges (slices) within AFF4 data objects. This is useful for a number of applications, including the annotation of content in TCP streams with the time of transmission, documenting provenance-related features of the derivation of an analytic product (e.g., file metadata), and describing the piecewise hash value of a chunk of an image.

An AFF4 slice provides the means to refer to a subrange within a URI. It is expressed by specifying the range within the fragment component of the URI:

```
URI#[offset:length]
```

The URI is a regular AFF4 Stream URI. The offset is a number that indicates the byte offset within the stream address range and the length is the number of bytes from the offset. For example, the slice URI:

```
urn:aff4:195bdf58-1bc9-4ba4-9a9c-f1c312673fbf#[512,128]
```

represents the address range from offset 512 to 640 in the URI.

```
urn:aff4:195bdf58-1bc9-4ba4-9a9c-f1c312673fbf
```

Note that the slice URI corresponds to an address range. In cases where the address range is backed by a stream of actual data, it also serves as a surrogate for the data within the bounds of the range. In the case where there is no corresponding data, it is a surrogate for the absence of data (i.e., an address space hole).

An example of a slice URI is demonstrated using a prototype piecewise hashing tool. Consider the following slice URI:

```
urn:aff4:f37648c1#[0,2048] aff4:hash "c35f2ba345"^^aff4:sha256
```

The interpretation of the slice URI (which has been truncated for presentation) is that the content of the byte range from 0 to 2048 of the stream `urn:aff4:f37648c1` has a SHA-256 value of `c35f2ba345`.

10.2 Representing Discontiguous Data

With the exception of the original AFF, existing forensic formats do not provide support for accurately imaging discontiguous data sources. Examples of discontiguous data sources include disks with read errors in certain sectors and the physical and virtual memory of computers with address space holes. For reasons of accuracy and completeness, it is important that the evidence container identifies areas of the Stream address space where there is no corresponding data, and potentially, the reason for the absence of data.

AFF4 provides a general solution to this problem by refining the semantics of the Map abstraction. Whereas the Map abstraction initially required that the target be a URI that resolves to a Stream or a Map, the refinement additionally allows the inclusion of a specially-defined URI as a target. Such a URI may indicate the characteristics of a byte range.

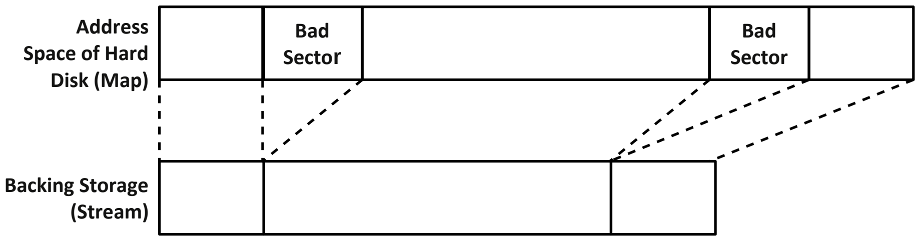


Figure 2. Address space mapping of a discontiguous evidence source.

For example, consider the imaging of a hard disk with a bad sector. We define a Map to represent the address space of the original evidence device and record the data content in a regular Stream. The Map records correspondences between the data content in the original device and the Stream. Figure 2 illustrates the mapping between the address space of an acquired hard drive with read errors and the underlying storage Stream. Note that the backing store stores valid data back to back, while the Map provides a view of the data with missing data represented as holes.

The address discontinuity that corresponds to the read error is given the target URI `aff4:UnknownData`. The following listing shows a Map segment that describes a discontinuity in this manner:

```
0,0,urn:aff4:da0d1948-846f-491d-8183-34ae691e8293
4096,0,http://libaff.org/2009/aff4#UnknownData
8192,4096,urn:aff4:da0d1948-846f-491d-8183-34ae691e8293
```

11. Representing Data Patterns

Storage media commonly comes from the factory with the data content of every byte set to zero. With hard drives rapidly increasing in capacity, it is often the case that large runs of data within images contain zeros ("zero data runs"). The AFF4 format reduces the storage impact of zero data runs by compressing the runs. However, due to the uniform chunking method, there is the potential for considerable repetition in sources containing large numbers of zero data runs.

Known data runs may be expressed in AFF4 using the slice URI and Map facilities. For zero runs, we define a special purpose URI with semantics similar to the UNIX `/dev/zero` to represent a Stream containing an unbounded number of zeros. This URI is `http://afflib.org/2009 /aff4#ZeroFilledByteRange`.

Of relevance to referencing common data patterns is recent work related to Teleporter [16], which explores the efficient transmission of forensic images by sending certain data runs of the image with reference to a standard corpus of files. Thus, in the case of a common file such as `ntfs.dll`, Teleporter would transport a fingerprint of the file rather than the entire data content. The receiver would then reconstitute the data content of the file from local sources.

12. Conclusions

The refinements made to the information and data models of the AFF4 evidence container format address the accurate representation of discontiguous data, help describe the provenance of stored evidence, and support authentication and non-repudiation of data and information by cryptographic signing.

Prototype implementations of AFF4 have been written in C, Python and Java. Our future research will integrate AFF4 with the PyFlag network forensic environment, the Volatility volatile memory analysis framework and the Sleuth Kit filesystem analysis tool. Also, it will ensure that AFF4 provides backward compatibility with the raw, EWF and AFF1 evidence container formats.

References

[1] M. Bartel, J. Boyer, B. Fox, B. LaMacchia and E. Simon, XML-Signature Syntax and Processing, World Wide Web Consortium, Cambridge, Massachusetts (www.w3.org/TR/xmldsig-core), 2009.

[2] D. Beckett and T. Berners-Lee, Turtle: Terse RDF Tripe Language, World Wide Web Consortium, Cambridge, Massachusetts (www .w3.org/TeamSubmission/turtle), 2008.

[3] T. Berners-Lee, R. Fielding and L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396 (www.ietf.org/rfc/rfc 2396.txt), 1998.

[4] B. Carrier, Defining digital forensic examination and analysis tools using abstraction layers, *International Journal of Digital Evidence*, vol. 1(4), 2003.

[5] J. Carroll, Signing RDF graphs, *Proceedings of the Second International Semantic Web Conference*, pp. 369–384, 2003.

[6] J. Carroll, C. Bizer, P. Hayes and P. Stickler, Named graphs, provenance and trust, *Proceedings of the Fourteenth International Conference on the World Wide Web*, pp. 613–622, 2005.

[7] J. Carroll and P. Stickler, TriX: RDF Triples in XML, Technical Report HPL-2003-268, HP Labs, Palo Alto, California (www.hpl.hp .com/techreports/2004/HPL-2004-56.pdf), 2004.

[8] M. Cohen, PyFlag: An advanced network forensic framework, *Digital Investigation*, vol. 5(S1), pp. S112–S120, 2008.

[9] M. Cohen, S. Garfinkel and B. Schatz, Extending the Advanced Forensic Format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow, *Digital Evidence*, vol. 6(S1), pp. S57–S68, 2009.

[10] S. Garfinkel, Providing cryptographic security and evidentiary chain-of-custody with the Advanced Forensic Format, library and tools, *International Journal of Digital Crime and Forensics*, vol. 1(1), pp. 1–28, 2009.

[11] S. Garfinkel D. Malan, K. Dubec, C. Stevens and C. Pham, Advanced Forensic Format: An open, extensible format for disk imaging, in *Advances in Digital Forensics II*, M. Olivier and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 13–27, 2006.

[12] R. Meijer, The Carve Path Zero Storage Library and Filesystem (ocfa.sourceforge.net/libcarvpath), 2006.

[13] R. Moats, URN Syntax, RFC 2141 (www.ietf.org/rfc/rfc2141.txt), 1997.

[14] B. Schatz and A. Clark, An information architecture for digital evidence integration, *Proceedings of the AusCERT Asia Pacific Information Technology Security Conference*, pp. 15–29, 2006.

[15] P. Turner, Unification of digital evidence from disparate sources (digital evidence bags), *Digital Investigation*, vol. 2(3), pp. 223–228, 2005.

[16] K. Watkins, M. McWhorte, J. Long and B. Hill, Teleporter: An analytically and forensically sound duplicate transfer system, *Digital Investigation*, vol. 6(S1), pp. S43–S47, 2009.

[17] World Wide Web Consortium, RDF/XML Syntax Specification (Revised), Cambridge, Massachusetts (www.w3.org/TR/REC-rdf-syntax), 2004.

Chapter 17

VIRTUAL EXPANSION OF RAINBOW TABLES

Vrizlynn Thing

Abstract Password recovery tools are often used in digital forensic investigations to obtain the passwords that are used by suspects to encrypt potential evidentiary data. This paper presents a new method for deterministically generating and efficiently storing password recovery tables. The method, which involves the virtual expansion of rainbow tables, achieves improvements of 16.92% to 28.15% in the password recovery success rate compared with the original rainbow table method. Experimental results indicate that the improvements are achieved with the same computational complexity and storage requirements as the original rainbow table method.

Keywords: Password recovery, cryptanalysis, rainbow table, time-memory trade-off

1. Introduction

Password protection of potential digital evidence by suspects makes investigative work more complex and time consuming. Traditional password recovery techniques include password guessing, dictionary attacks and brute force attacks.

A password guessing technique attempts easily-formed and common passwords such as "qwerty" and "password." These passwords could be based on a user's personal information or a fuzzy index of words on the user's storage media. A statistical analysis of 28,000 passwords revealed that 16% of the users relied on their first names as passwords and 14% relied on "easy-to-remember" keyboard combinations [2]. Therefore, the password guessing method can be effective in cases where users are willing to compromise security for convenience.

A password dictionary attack attempts to match the hash values of dictionary words with the stored password hash value. Well-known tools that employ this technique include Cain and Abel [7], John the Ripper [12] and LCP [6].

In a brute force cryptanalytic attack, the hash value of each unique combination of password characters is compared with the password hash value until a match is found. Although such an attack is extremely time consuming, the password is recovered eventually. Cain and Abel, John the Ripper and LCP also support brute force attacks.

Traditional password recovery techniques are losing their effectiveness as suspects use stronger passwords to protect their data. Hellman [5] introduced a method that trades off the computational time and storage space needed to make a hash-to-plaintext recovery. This method can retrieve Windows login passwords as well as passwords used in other applications (e.g., Adobe Acrobat) that employ the LM and NTLM hash algorithms [15]. Also, the method supports encryption key recovery for Microsoft Word and Excel documents. Passwords encrypted with hashing algorithms such as MD5 [13], SHA-2 [9] and Ripemd-160 [4] can also be recovered using this method. In general, Hellman's method is applicable to searching for solutions to knapsack and discrete logarithm problems.

Oechslin [11] proposed a cryptanalytic time-memory trade-off method that is based on Hellman's password recovery method. This "rainbow table" method generates password recovery tables with higher efficiency because it employs multiple reduction functions to reduce the probability of collisions in a single table.

Thing and Ying [14] enhanced the rainbow table method via initial chain generation. In this technique, a plaintext value is chosen and its hash value is computed by applying the password hash algorithm. Based on this hash value, other hashes are computed; these form the branches of the initial plaintext value. Multiple blocks are created from the different initial plaintext values. The final values of the chains in the blocks are stored with the single initial plaintext value in each block.

The rainbow table method is widely used for password recovery. Products that use pre-computed rainbow tables include RainbowCrack [16] and Ophcrack [10]. RainbowCrack is an implementation of Oechslin's method that supports hash algorithms such as LM, NTLM, MD5 and SHA-1 [8]. Ophcrack also implements the rainbow table method, but it only supports the LM and NTLM hash algorithms. Rainbow tables are also used in several popular commercial tools such as AccessData's Password Recovery Toolkit [1] to perform efficient and effective password recovery.

This paper proposes the virtual expansion of rainbow tables (VERT) method, a new time-memory trade-off technique that relies on a pre-computed table structure. Comparisons with the original rainbow table method [11] and the enhanced rainbow table method [14] demonstrate higher password recovery success rates of 16.92% to 28.15% without corresponding increases in computational complexity and storage space.

2. Related Work

In 1980, Hellman [5] conceived of a general time-memory trade-off hash-to-plaintext recovery method. We begin by describing Hellman's algorithm in the context of password recovery.

Let X be the plaintext password and Y be its corresponding stored hash value. Given Y, it is necessary to find X that satisfies the equation $h(X) = Y$ where h is the known hash function. However, finding $X = h^{-1}(Y)$ is not feasible because the hash values are computed using one-way functions for which the reversal function h^{-1} is unknown.

To solve this problem, Hellman suggested applying alternate hashing and reduction operations to plaintext values in order to generate a pre-computed table. For example, given a 7-character password (using the English alphabet), the MD-5 hash algorithm is used to compute the corresponding 128-bit hash value. Using a reduction function such as $H \bmod 26^7$, where H is the hash value converted to decimal digits, the resulting values are distributed in a best-effort uniform manner.

For an initial plaintext password of `abcdefg`, the binary hash output could be:

```
00000000,00000000,00000000,00000000,00000000,00000000,
00000000,00000000,10000000,00000000,00000000,00000000,
00000000,00000000,00000000,00000001.
```

Therefore, $H = 9223372036854775809$. The reduction function converts this value to `3665127553`, which corresponds to the plaintext representation of `lwmkgij`. This is computed as:

$$11(26^6) + 22(26^5) + 12(26^4) + 10(25^3) + 6(26^2) + 8(26^1) + 9(26^0).$$

In table generation, the hashing and reduction operations are repeatedly performed on different initial plaintext values to produce different rows or chains. The pre-defined multiple rounds of hashing and reduction operations in each chain increase the length of the chains and the table contents; this contributes to higher computational overhead during password recovery. The reason is that the initial and final plaintext values (i.e., "heads" and "tails") of the chains are the only elements stored in the table. The recovery of passwords "residing" in the intermediate columns requires computations to regenerate the plaintext-hash pairs.

The password recovery success rate depends on the size of the pre-computed table. A larger pre-computed table contains more plaintext-hash pairs, which increases the password recovery success rate. However, the generation of the intermediate chain columns may result in collisions of elements in the table. Collisions cause the chains to merge and reduce the number of (distinct) plaintext-hash pairs. This decreases the password recovery success rate.

To increase the success rate, Hellman proposed using multiple tables where each table has a different reduction function. If $P(t)$ is the success rate using t tables, then $P(t) = 1 - (1 - P(1))^t$, which is an increasing function of t. Hence, introducing more tables increases the password recovery success rate but increases the computational complexity and storage requirements.

A plaintext password is recovered from its hash value by performing a reduction operation on the hash value. Next, a search is conducted for a match of the computed plaintext value with a final value in the table. If a match is not found, the hashing and reduction operations are performed on the computed plaintext value to obtain a new plaintext value for another round of search. The maximum number of rounds of hashing, reduction and searching operations depend on the pre-defined chain length.

Rivest [3] suggested using distinguished points as end points when generating the chains. Distinguished points are values that satisfy a given criterion (e.g., the first q bits are 0). In this method, chains are not generated with a fixed length; instead, they terminate upon reaching a pre-defined distinguished point.

This method decreases the number of memory lookups compared with Hellman's method and is capable of loop detection. If a distinguished point is not reached after a large number of operations, it is assumed that the chain contains a loop and is discarded. One limitation is that the chains will merge if there are collisions within the same table. The variable lengths of the chains also increase the number of false alarms. Additional computations are needed to rectify these errors.

Oechslin [11] proposed a new table structure that reduces the probability of chains merging. The method uses a different reduction function to generate the elements in each chain column. Therefore, chains merge only when a collision occurs in the same column. For m chains of length n, the total number of possible chain merges (i.e., when two similar elements appear in the same column) is:

$$\frac{nm(m-1)}{2}.$$

The total number of possible pairs not in the same chain column is:

$$\frac{n^2m(m-1)}{2}.$$

Therefore, given a collision in the table, the probability that there is a chain merge is:

$$\frac{\frac{nm(m-1)}{2}}{\frac{n^2m(m-1)}{2}} = \frac{1}{n}.$$

Oechslin showed that the measured coverage in a single "rainbow" is 78.8% compared with 75.8% in a classical rainbow table constructed using distinguished points. In addition, the search effort is reduced, which contributes to an improvement in performance.

Thing and Ying [14] proposed an enhancement to the rainbow table method. The enhancement is achieved through initial chain generation by systematically manipulating the initial hash values based on an adjustable parameter k. A plaintext value is chosen and its hash value is computed using the password hash algorithm. The resulting hash value H is used to compute:

$$(H+1)\ mod\ 2^j, (H+2)\ mod\ 2^j, \ldots, (H+k)\ mod\ 2^j,$$

where j is the number of bits in the hash output. These hash values form the branches of the initial plaintext value. Alternate hashing and reduction operations are then applied to these branches. The resulting extended chains form a block. The final values of the chains in each block are stored with the single initial plaintext value. The same operations are performed on all the other initial plaintext values. These sets of initial and final values form the new pre-computed table.

Instead of storing all the initial and final plaintext values as pairs as in the rainbow table, an initial plaintext value is stored with multiple output plaintext values. This results in significant storage space savings compared with the rainbow table method. In particular, for the same storage space as the rainbow table method, the enhanced method yields 13.28% to 19.14% improvement in the total number of distinct plaintext-hash pairs generated. However, some passwords "residing" in the first column will not be recovered because they are not stored in the table. This limits the search to $n-1$ chain columns instead of n columns.

3. Rainbow Table Method

We use the following notation in our discussion of the rainbow table method and its extensions: x_i denotes the initial value of a chain; y_i

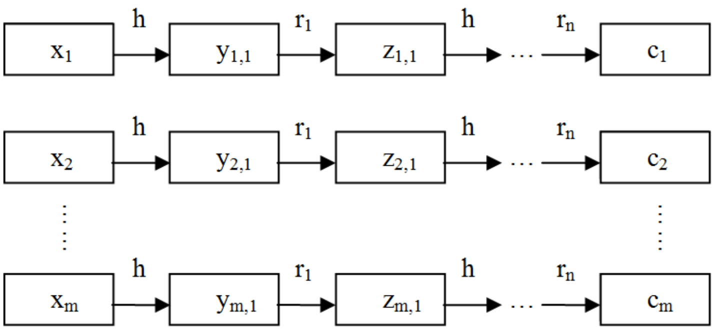


Figure 1. Rainbow table.

denotes the hash value of a password; z_i denotes the reduced value of a hash; c_i denotes the final value of a chain; h denotes a hash function; r_i denotes a reduction function; n denotes the number of reduction functions or chain length; and m denotes the number of chains.

In order to generate a rainbow table, it is necessary to specify reduction functions $r_1, r_2, \ldots, r_n$ that convert hash values y_i to plaintext values z_i. A large set of plaintext values $x_1, x_2, \ldots, x_m$ are then chosen as the initial values of the table. As shown in Figure 1, these plaintext values are alternately hashed using the password hashing algorithm and reduced using the reduction functions.

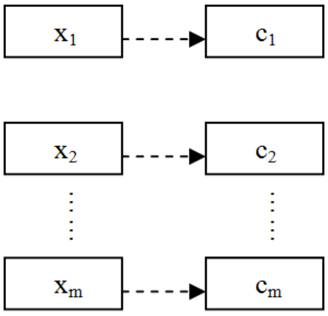


Figure 2. Stored values in a rainbow table.

A rainbow table is created by only storing the final values c_i along with their corresponding initial values x_i (Figure 2). A password is recovered by alternately applying reduction and hashing operations to the corresponding hash values until a value is obtained that matches one of the final values in the rainbow table.

Consider the situation where it is necessary to find the password corresponding to the password hash v. One round of the reduction operation is applied to the password hash v to obtain the plaintext value w. Next,

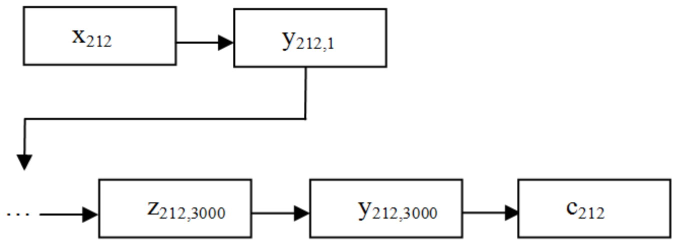


Figure 3. Password recovery example.

a search is performed and a match of w is found at the final value of the 212th chain in the rainbow table in Figure 3 (i.e., $w = c_{212}$). The chain is then computed from its initial value x_{212} until the password hash v is reached, which is equal to $y_{212,3000}$. The password is the plaintext value $z_{212,3000}$, which is before the password hash $y_{212,3000}$.

If no matching value is found, it is assumed that the particular password does not exist in the rainbow table and cannot be recovered. The password recovery success rate can be improved by increasing the number of reduction functions and chains, but this increases the computational complexity and storage requirements.

4. Virtual Expansion of Rainbow Tables Method

Our proposed virtual expansion of rainbow tables (VERT) method virtually expands the rainbow table contents while maintaining the storage space requirements of the original rainbow table method. In VERT, the character set is first remapped to numerical equivalent values using the VERT mapping table.

An example VERT mapping for the alphanumeric character set is shown in Table 1. For a 7-character password, the initial plaintext value in the first chain of the VERT table is selected to be `0000000` and the initial plaintext value in the last chain is selected to be `zzzzzzz`. The initial plaintext values of the remaining chains are chosen from the rest of the password space based on evenly-distributed gaps. The gap size depends on the storage contraint. For example, if the storage space is only sufficient to store four plaintext values, the gap size is:

$$(36^7 - 1)/(4 - 1) = 26121388032$$

after rounding up to the next integer. The four initial plaintext values are `0000000`, `c000001` (computed from $26121388032 = 12(36^6)+1(36^0)$), `n000000` (computed from $2(26121388032) = 52242776064 = 24(36^6)$) and `zzzzzzz`. These initial plaintext values are not stored in the VERT

Table 1. VERT mapping table.

Character Set	Numerical Equivalent
0	0
1	1
2	2
...	...
a	10
b	11
...	...
z	35

table. Instead, only their final plaintext values are stored. Therefore, VERT provides a 100% increase in the number of chains compared with the original rainbow table [11]. Also, it supports password recovery to the first column unlike the enhanced rainbow table method [14].

The VERT method also incorporates an efficient storage mechanism that can support a larger table while using the same amount of storage as the original rainbow table. As seen in Table 1, each character in the VERT mapping table can be represented by six bits. Therefore, the final plaintext values of the chains are converted to their numerical representations before storage. Because eight bits of storage are required for each plaintext character in a rainbow table, the VERT method provides an additional storage conservation of up to 25% and an additional 33.33% increase in the number of chains without increasing the storage requirements.

5. Theoretical Analysis

This section presents a theoretical analysis and comparison of the performance of the VERT, original rainbow table and enhanced rainbow table methods.

5.1 Success Rate without Collisions

The password recovery success rate depends on the number of distinct plaintext-hash pairs generated in the chains, which, in turn, depends on the total number of plaintext-hash pairs generated. First, we perform the analysis ignoring element collisions. Next, we perform an analysis based on the number of distinct pairs and evaluate the effect on the password recovery success rate.

If there are m rainbow chains, each with n reduction functions and requiring storage for two plaintext values (initial value and final value of a chain), then a rainbow table has to store a total of $2m$ plaintext values. Thus, the number of plaintext-hash pairs is mn.

In the case of the enhanced rainbow table method, optimal performance is achieved when a single block is formed. Therefore, using the same storage space as in the original rainbow table, a total of $n(2m-1)$ plaintext-hash pairs can be generated. This is computed from $2m(n-1)-(n-2) \approx 2mn-n \approx n(2m-1)$ assuming that $n \gg 2$.

The VERT method does not store the input plaintext values. Therefore, a VERT table can generate a total of $2(1.3333m) = 2.67m$ chains using the same storage space as the original rainbow table. Therefore, a VERT table would have $2.67mn$ plaintext-hash pairs. Compared with the original and enhanced rainbow table methods, this translates to 167% and 33% increases in the password recovery success rate, respectively. This success rate is based on the additional plaintext-hash pairs that are generated.

5.2 Success Rate with Distinct Pairs

The password recover success rate computed above ignores collisions. Because the collision probability increases with the size of a rainbow table, ignoring collisions is reasonable only for very small rainbow tables. We compute a more realistic password recovery success rate based on collisions with the distinct plaintext-hash pairs that are generated.

Our analysis assumes that storage space exists for m plaintext values. The password recovery success rate is computed based only on the distinct plaintext-hash pairs. The same number of reduction functions n is used for the original rainbow table, enhanced rainbow table and VERT methods.

Let N be the password space that comprises all possible plaintext passwords and let m_i be the number of distinct plaintext-hash pairs in the i^{th} column of the original rainbow table. Then, m_i and m_{i+1} satisfy the recurrence relation:

$$m_{i+1} = N(1-(1-\frac{1}{N})^{m_i})$$

where $m_1 = m$. Thus, the probability of successful password recovery for the original rainbow table method is:

$$P(M) = 1-(1-\frac{m_1}{N})(1-\frac{m_2}{N})\dots(1-\frac{m_n}{N}).$$

Table 2. Success rate based on distinct plaintext-hash pairs.

Storage Size (m)	Original Rainbow Table	Enhanced Rainbow Table	VERT Table
10×10^6	45.07%	65.33%	73.22%
15×10^6	56.94%	76.15%	82.62%
20×10^6	65.33%	82.60%	87.82%
25×10^6	71.50%	86.74%	91.00%
30×10^6	76.15%	89.57%	93.07%

In the case of the enhanced rainbow table method, let s_i be the number of distinct plaintext-hash pairs in the i^{th} column. Then, s_i and s_{i+1} satisfy the following recurrence relation:

$$s_{i+1} = N(1 - (1 - \frac{1}{N})^{s_i}) \ \forall i > 1; s_1 = 1; s_2 = 2m - 1.$$

Thus, the probability of successful password recovery for the original rainbow table method is:

$$P(S) = 1 - (1 - \frac{s_1}{N})(1 - \frac{s_2}{N}) \ldots (1 - \frac{s_n}{N}).$$

In the case of the VERT method, let v_i be the number of distinct plaintext-hash pairs in the i^{th} column of the VERT table. Then, v_i and v_{i+1} satisfy the recurrence relation:

$$v_{i+1} = N(1 - (1 - \frac{1}{N})^{v_i})$$

where $v_1 = 2.67m$. Thus, the probability of successful password recovery for the VERT method is:

$$P(V) = 1 - (1 - \frac{v_1}{N})(1 - \frac{v_2}{N}) \ldots (1 - \frac{v_n}{N}).$$

Note that $v_i > m_i$ for all $i \geq 1$. Thus, $P(V) > P(M)$. In addition, $P(V) > P(S)$ since $v_i > s_i$ for all $i \geq 1$.

The password recovery success rates for the original and enhanced rainbow tables and for the VERT tables for different numbers of stored plaintext values (i.e., storage space) are computed based on the equations presented above. The results are presented in Table 2. The common parameters used in the methods are: (i) number of reduction functions (n): 5,700; (ii) character set: alphanumeric; (iii) plaintext/password length: 1-7 characters; and (iv) storage space (m): same for all methods.

Table 2 shows that the VERT method yields password recovery success rate improvements ranging from 16.92% for storage size $m = 30 \times 10^6$ to 28.15% for storage size $m = 10 \times 10^6$. When compared with the enhanced rainbow table method, the VERT method provides password recovery success rate improvements ranging from 3.50% for $m = 30 \times 10^6$ to 7.89% for $m = 10 \times 10^6$. The results show that even when collisions are considered, the VERT method offers substantial improvements in performance. Note also that the storage size constraint impacts the original rainbow table method much more significantly than the enhanced rainbow table and VERT methods.

6. Experimental Results

This section compares the results obtained with the VERT method and those obtained using RainbowCrack (source code version 1.2) [16].

6.1 Distinct Passwords and Success Rate

To evaluate the password recovery success rate when considering distinct pairs, we made a slight modification to RainbowCrack to log all the plaintext passwords (i.e., the stored initial and final columns as well as the intermediate chain columns). This logging was also performed for the VERT method. We also implemented scripts to detect collisions and count the distinct passwords in the logs.

The experiments were conducted using the following common parameters: (i) number of reduction functions (n): 3,000; (ii) character set: lower case alpha; (iii) plaintext/password length: 1-7 characters; and (iv) storage space (m): 10^6. For fixed $m = 10^6$, this translates to 10^6 chains for the RainbowCrack method and 2.67×10^6 chains for the VERT method.

The theoretical password recovery success rates for RainbowCrack and VERT when considering distinct pairs are 28.13% and 54.31%, respectively. In the experiments, the total plaintext-hash pairs generated were 3×10^9 by RainbowCrack and 8.01×10^9 by VERT. The total plaintext space was 8,353,082,582. A total of 2,349,122,955 distinct passwords were identified among the 3×10^9 plaintext passwords generated by RainbowCrack, corresponding to an actual password recovery success rate of 28.12%. On the other hand, VERT generated 4,536,258,880 distinct passwords, yielding an actual password recovery success rate of 54.31%. Thus, the experimental results match the theoretical results.

Note that the experiments conducted are preliminary in nature due to the scale of collision detection and distinct password count computations. Additional experiments will be performed to study the impact of collisions for larger numbers of chains and reduction functions.

6.2 Computational Complexity

A password is computed by alternatively applying the reduction and hashing operations to the password hash value. However, VERT requires an additional final step of processing (numerical representation conversion) on the last computed plaintext value before it is stored in the table. This conversion is a simple operation; thus, it incurs insignificant computational overhead compared with hashing and reduction.

We conducted experiments on an Intel P4 3.06 GHz system to compute the time taken to perform: (i) random value generation for the initial column for RainbowCrack; (ii) deterministic value generation for the initial column for VERT, (iii) reduction and MD5 hashing operations for RainbowCrack and VERT (used for intermediate column processing); and (iv) single conversion of the final plaintext password to its numerical representation. A total of 10^9 rounds were performed for each operation and the average time for each round was computed. The average computation times were: (i) 1,656 ns; (ii) 7 ns; (iii) 644 ns; and (iv) 211 ns. Note that the final conversion operation only incurs an additional 211 ns for each chain. The initial value generation speed is greatly enhanced in VERT. The total time taken to generate the initial values for RainbowCrack is 1,656m ns while the time taken for VERT is $(2.67 \times 7)m$ = 18.69m ns. However, the main computational overhead is due to the reduction and hashing operations, which require a total of 644mn ns.

7. Conclusions

The novelty of the VERT method lies in the virtual expansion of the pre-computed tables, which increases the password recovery success rate while limiting the storage requirements. Compared with the original rainbow table method, the VERT method increases the password recovery success rate by 16.92% to 28.15% for the distinct pairs comparison while considering collision effects. The VERT method also shows an improvement in the password recovery success rate compared with the enhanced rainbow table method. In particular, the VERT method yields up to 33% improvement for the total plaintext-hash pairs comparison and 3.5% to 7.89% improvement for the distinct pairs comparison. Note also that it is possible to trade-off storage conservation in favor of high password recovery success rates for longer passwords (i.e., larger password spaces).

Our future work related to the VERT method will analyze collisions and password recovery success rates in larger tables. Additionally, we plan to investigate improvements in the password recovery time achieved by reducing the number of columns while maintaining the same storage requirements and password recovery success rates as the original and enhanced rainbow table methods.

References

[1] AccessData, Decryption tools, Lindon, Utah (www.accessdata.com /decryptionTool.html).

[2] Agence France-Presse, Favorite passwords: "1234" and "password," Paris, France, February 11, 2009.

[3] D. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, Massachusetts, 1982.

[4] H. Dobbertin, A. Bosselaers and B. Preneel, Ripemd-160: A strengthened version of RIPEMD, *Proceedings of the Third International Workshop on Fast Software Encryption*, pp. 71–82, 1996.

[5] M. Hellman, A cryptanalytic time-memory trade-off, *IEEE Transactions on Information Theory*, vol. 26(4), pp. 401–406, 1980.

[6] LCPSoft, LCP, Moscow, Russia (www.lcpsoft.com).

[7] M. Montoro, Cain and Abel (www.oxid.it/cain.html).

[8] National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standards Publication 180-1, Gaithersburg, Maryland, 1995.

[9] National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standards Publication 180-2, Gaithersburg, Maryland, 2002.

[10] Objectif Securite, Ophcrack, Gland, Switzerland (ophcrack.source forge.net).

[11] P. Oechslin, Making a faster cryptanalytic time-memory trade-off, *Proceedings of the Twenty-Third International Cryptology Conference*, pp. 617–630, 2003.

[12] Openwall Project, John the Ripper password cracker (www.open wall.com/john).

[13] R. Rivest, The MD5 Message-Digest Algorithm, IETF RFC 1321, 1992.

[14] V. Thing and H. Ying, A novel time-memory trade-off method for password recovery, *Digital Investigation*, vol. 6(S1), pp. S114–S120, 2009.

[15] D. Todorov, *Mechanics of User Identification and Authentication: Fundamentals of Identity Management*, Auerbach Publications, Boca Raton, Florida, 2007.

[16] S. Zhu, RainbowCrack: The time-memory trade-off hash cracker (project-rainbowcrack.com).

Chapter 18

DIGITAL WATERMARKING OF VIRTUAL MACHINE IMAGES

Kumiko Tadano, Masahiro Kawato, Ryo Furukawa, Fumio Machida and Yoshiharu Maeno

Abstract The widespread use of server and desktop virtualization technologies increases the likelihood of unauthorized and uncontrolled distribution of virtual machine (VM) images that contain proprietary software. This paper attempts to address this issue using a platform-independent digital watermarking scheme applicable to a variety of VM images. The scheme embeds a watermark in the form of files in a VM image; the watermarked VM image is identified based on the embedded files. To reduce the possibility of discovery by an attacker, the names of the embedded files are very similar to the names of pre-existing files in the VM image. Experiments indicate that the approach is fast and accurate, with average turnaround times of 24.001 seconds and 7.549 seconds for watermark generation and detection, respectively.

Keywords: Digital watermarking, virtual machine images

1. Introduction

In modern enterprise computing there is a growing trend toward consolidating virtual machines (VMs) in the server and client sides to reduce costs and enhance portability and security. Such environments require the means to export VM images to test software and to backup VMs. For example, Amazon's Elastic Compute Cloud [1] and Simple Storage Service [2] provide the command `ec2-download-bundle` to download VM images from a data center to local computers [3]. Unfortunately, this technology also facilitates the unauthorized dissemination of VM images containing proprietary software.

One approach for addressing this issue is to use host-based intrusion detection systems such as TripWire, AIDE and XenFIT [8]. These systems monitor unauthorized file system changes to detect malicious activity involving VMs. However, it is difficult to identify VM images after they have been distributed outside of an enterprise network (e.g., using peer-to-peer file sharing software).

Another approach is to implement strict access control and copy control of VM images. However, these controls often hinder the legitimate use of VMs.

Therefore, it is necessary to address two issues: (i) identify VM images even after they have been illegally distributed; and (ii) facilitate legitimate use of VM images. Digital watermarking of VM images can address both these issues. However, as we discuss below, watermarking techniques used for audio and video files are not applicable to VM images.

Digital watermarking technologies typically modify redundant or unused digital content (e.g., inaudible frequency ranges for audio files) to embed watermarks. In the case of audio and video files, modifying the original information for digital watermarking produces imperceptible effects when playing the files [7]. In contrast, modifying a VM image can cause boot failures when the watermarked image is executed. Additionally, data on the watermarked VM is frequently changed because of software updates, logging, etc. Unlike an audio or video file whose content is unchanged, a VM image is essentially variable; consequently, in order to identify the VM image, the image has to be watermarked after every change.

Data hiding techniques [5] can be employed for watermarking VM images. Data can be hidden in various locations:

- Areas marked as not in use by the partition table.
- Extended file attributes such as alternate data streams.
- Unused portions of the last data units of files (slack space).
- Reserved i-nodes that are not used by the operating system.
- Portions that are excluded during consistency checking of a journaling file system.
- Files hidden via steganography using special file system drivers [6].

A major deficiency of existing data hiding techniques is the lack of platform-independence: the techniques work on specific file systems (e.g., NTFS), operating systems (e.g., Red Hat Linux) and OS kernel versions (e.g., Linux 2.2.x). It is difficult, if not impossible, to apply these techniques to VMs in heterogeneous environments where multiple file systems, operating systems and OS kernel versions are used.

To address this issue, we propose a digital watermarking scheme for VM images that is independent of file systems, file formats, operating systems, OS kernel versions and hardware. The scheme embeds a watermark derived from the names of the files present in a VM.

2. Basic Concepts

Our digital watermarking scheme for VM images uses file names as a watermark, not the contents of the files. A unique identifier is created for each VM image and a watermark corresponding to the identifier is embedded in the form of files in the file system of the VM image. These embedded files are called "watermark fragment files."

If the names of the watermark fragment files are randomly created, they would be readily distinguishable from the names of the pre-existing files on the VM, enabling an attacker to identify and subsequently remove the watermark fragment files. Thus, the watermarking scheme makes the fragment files difficult to detect by creating fragment files with names that are similar to pre-existing files in the VM and embedding the files in random directories in the VM. A secure database is used to store the identifier of each VM image and the corresponding watermark fragment files.

3. Design and Implementation

This section describes the design and implementation of the digital watermarking scheme.

3.1 System Components

The system has three components: (i) a watermark generator; (ii) a watermark detector; and (iii) a watermark information database. Figure 1 illustrates the watermark generation and detection processes.

- **Watermark Generator:** The watermark generator employs the names of the watermark fragment files as the watermark for a VM image. It creates a watermarked VM image by embedding the watermark fragment files in the VM image and stores the generated file names in the watermark information database. The VM image to be watermarked is created by copying the template VM image file. Since the template VM images are used as the original data for watermarking, these images should be securely managed. The reason is that an attacker who obtains the VM image template

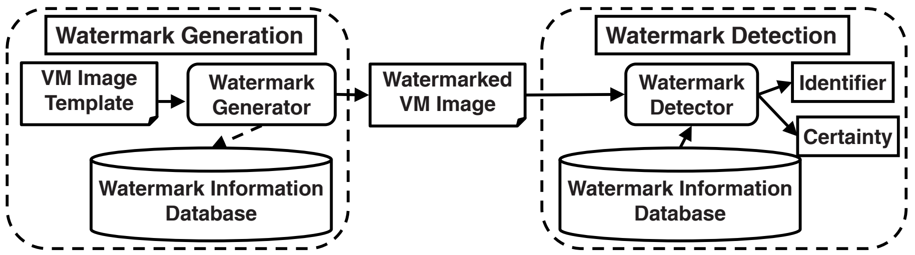
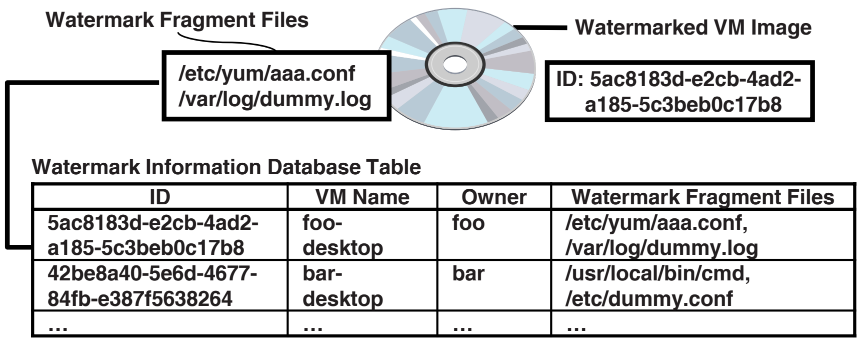


Figure 1. Watermark generation and detection.

would be able to compute the watermark fragment files as the difference between the template VM image and the watermarked VM image files. Details of the watermark generation algorithm are presented in Section 3.2.



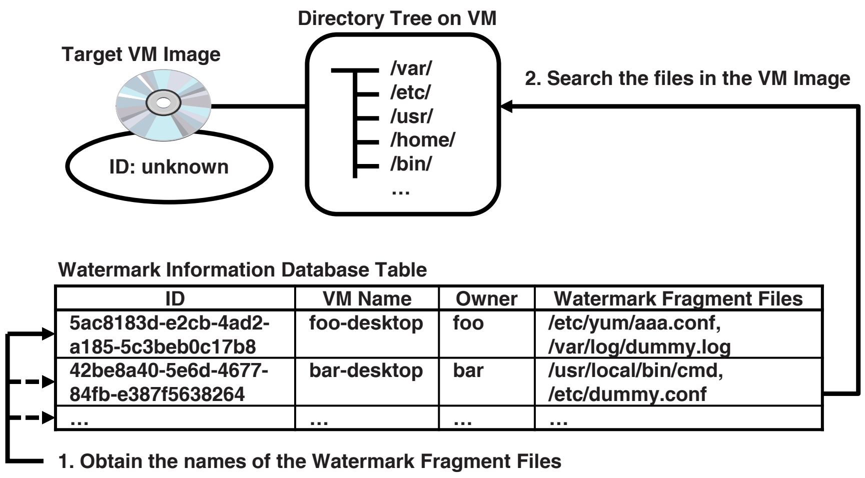
ID	VM Name	Owner	Watermark Fragment Files
5ac8183d-e2cb-4ad2-a185-5c3beb0c17b8	foo-desktop	foo	/etc/yum/aaa.conf, /var/log/dummy.log
42be8a40-5e6d-4677-84fb-e387f5638264	bar-desktop	bar	/usr/local/bin/cmd, /etc/dummy.conf
...	...	...	...

Figure 2. Sample records in the watermark information database.

- **Watermark Information Database:** The watermark information database (WI-DB) stores the identifier of each VM image and its attributes. The identifier for a VM image is a universally unique identifier (UUID). The attributes include the names of the watermark fragment files, VM owner, etc. (Figure 2).

  Only authorized users should be permitted to read and write WI-DB records. An attacker with read/write access could alter WI-DB records or identify and delete the watermark fragment files embedded in a VM image.

- **Watermark Detector:** The watermark detector identifies a VM using the watermark fragment files embedded in the target VM image based on WI-DB records. When an unauthorized leak of a VM image is suspected, the administrator may perform the detection process to identify the VM image and its owner.


Directory Tree on VM
Target VM Image
ID: unknown
/var/
/etc/
/usr/
/home/
/bin/
...
2. Search the files in the VM Image
Watermark Information Database Table
1. Obtain the names of the Watermark Fragment Files

ID	VM Name	Owner	Watermark Fragment Files
5ac8183d-e2cb-4ad2-a185-5c3beb0c17b8	foo-desktop	foo	/etc/yum/aaa.conf, /var/log/dummy.log
42be8a40-5e6d-4677-84fb-e387f5638264	bar-desktop	bar	/usr/local/bin/cmd, /etc/dummy.conf
...	...	...	...

Figure 3. Watermark detection.

Figure 3 illustrates the detection process. If the identifier of the target VM can be guessed, the watermark generator looks up the corresponding WI-DB record. Otherwise, the watermark detector compares the watermark fragment file names in WI-DB records with those in the target VM image. The watermark generator then outputs a certainty value of the identifier of the target VM image. The certainty value is computed as the percentage of the names of the watermark fragment files in the WI-DB record that match file names in the VM image. Note that some watermark fragment files may have been deleted or renamed on purpose or by accident.

3.2 Watermark Generation Algorithm

This section briefly describes the process of generating the watermark fragment files for the target VM image. The algorithm has five steps.

- **Step 1:** Input the parameters required for watermark generation.
- **Step 2:** Number the directories in the target VM image in dictionary order.
- **Step 3:** Select the directories to embed watermark fragment files.
- **Step 4:** Generate the names of the watermark fragment files to be created.
- **Step 5:** Generate the contents of the watermark fragment files for the directories selected in Step 3.

Table 1. Directory code list.

Number	Directory Name
1	`/`
2	`/etc/`
3	`/etc/init.d/`
4	`/etc/ldap/`
5	`/etc/skel/`
6	`/etc/ssh/`
7	`/etc/sysconfig/`
...	...

Step 1 (Input Parameters for Watermark Generation): The following input parameters are required for watermark generation:

- **Number of Watermark Fragment Files:** The larger the number of watermark fragment files, the more tamper-resistant is the watermark. However, this increases the time required to generate watermark fragment files.
- **Excluded Directories:** Some directories (e.g., temporary directories) should be excluded because files in these directories change frequently.
- **Excluded Suffixes:** These are suffixes that should be excluded from the names of the watermark fragment files.
- **Bit Length:** The bit length is a parameter used by the Blum-Blum-Shub (BBS) algorithm [4] to generate cryptographically-secure pseudo-random numbers. This parameter affects the computational time and security, and is set to 1024 bits in our experiments.

Step 2 (Number the Directories in the Target VM Image): The watermark generator searches all the directories in the target VM image recursively and generates a "directory code list" (Table 1). The excluded directories identified in Step 1 are not numbered. The directory code of a template VM image can be reused when new watermarked VM images are created from the same template VM image because they have the same directory tree structure. This reduces the processing time.

Step 3 (Select Directories to Embed Watermark Fragment Files): The watermark generator produces pseudo-random numbers for the watermark fragment files input in Step 1. The values of the random numbers are limited to the range of directory codes. The watermark generator extracts the names of directories corresponding to the generated random numbers using the directory code list generated in Step 2. The BBS pseudo-random number generation algorithm is used to make the random numbers difficult for an attacker to predict. Although this algorithm is computationally intensive, the experimental results presented in Section 4.2 indicate that the overall performance of the watermarking scheme is acceptable.

Step 4 (Generate the Names of Watermark Fragment Files): The watermark generator creates the names of the watermark fragment files that are embedded in the directories selected in Step 3. Portions of the names of pre-existing files in the directories are modified to create file names that are similar to those of the pre-existing files. A file name is generated according to the following steps.

- **Step 4.1:** Get the names of existing files in the target directory. Only regular files (not subdirectories, hidden files, etc.) whose suffixes are not to be excluded are retrieved.

- **Step 4.2:** Enumerate all the substrings corresponding to the file names. Each substring is obtained from the head (i.e., not including the suffix) of a file name retrieved in Step 4.1. The result of this step is the union of substrings for all the file names. For example, if the target directory has a file named `abc.txt`, the possible substrings are: `abc`, `ab` and `a`.

- **Step 4.3:** Compute "similarity groups" for each extracted substring. A similarity group is a group of files in the target directory that meets the following conditions: (i) the file name starts with the substring (prefix) generated in Step 4.2; (ii) all the files have the common suffix (e.g., `.txt`); and (iii) a similarity group contains at least two files. For example, if there are five files in a target directory {`s1.txt`, `s2.txt`, `s3.dat`, `s4.dat`, `s5.conf`} and `s` is the substring, then two similarity groups can be computed: {`s1.txt`, `s2.txt`} and {`s3.dat`, `s4.dat`}.

- **Step 4.4:** Compute the D_{sim} score for each similarity group:

  $$D_{sim} = c_1 * l_p + c_2 * n_f - c_3 * l_d$$

  where l_p is the length of the prefix of the similarity group; n_f is the number of files in the similarity group; l_d is the mean value of the difference between the length of the file name (excluding the suffix) and l_p in the similarity group; and c_1, c_2, c_3 are parameters that are set to one. The similarity group with the highest D_{sim} score is called the "prototype file group."

- **Step 4.5:** Create the name of the new watermark fragment file. One or more random letters are added to the tail of the prefix of the selected prototype file group. The length of the added letters (l_t) is the length of the file name randomly selected from the prototype file group (excluding the suffix and prefix). Finally, the suffix of the files in the prototype file group is added to the file name. For example, if the prefix is `sample` and the prototype file group is {`sample.dat`, `sample-bak.dat`} and if `sample-bak.dat` is selected, then $l_t = 4$ and a possible name is `sampledbha.dat`.

Step 5 (Generate the Content of the Watermark Fragment Files): The watermark generator creates the content and attributes corresponding to each watermark fragment file. The content of a watermark fragment file is a randomly-generated byte sequence. Attributes of a watermark fragment file are determined according to the following rules. For timestamps (creation/modification/access) and file size, the watermark generator assigns the mean values of the files in the prototype file group to the new watermark fragment file. For other attributes such as ownership and permission, the values corresponding to a randomly selected file in the prototype file group are used.

3.3 Implementation

The digital watermarking scheme was implemented in Java 1.5. The VMware Server virtualization software was employed. The command `vmware-mount.pl` was issued to the VMware Server to mount the file system on the VM image for embedding watermark fragment files. The VM image of any file system or operating system, including Windows and Linux, can be watermarked. The WI-DB was implemented using MySQL 5.0.

4. Experimental Setup and Results

This section describes the experimental setup used to evaluate the watermarking algorithm. Also, it presents the experimental results related to watermark generation and detection.

4.1 Experimental Setup

A test environment was created to evaluate the performance of the watermarking scheme. The environment included one physical host (Ta-

Table 2. Test environment.

Physical Host		
CPU	Pentium 4 1.73 GHz Processor	
Memory	1 GB RAM	
Host OS	Ubuntu Linux 8.04 Server	
VMM	VMware Server 2.0	
	Guest VMs	
Guest OS	**Ubuntu Linux 8.04 (JeOS edition)**	**CentOS 5.1 (default)**
VM Image Size (`.vmdk file`)	183 MB	2,750 MB
Directories	1,120	7,429

ble 2). Two template VM images of different sizes were used to clarify the impact of size on the watermarking scheme. The VM images used were an Ubuntu Linux 8.04 Server JeOS edition (minimum configuration for virtual appliances) and a default installation of CentOS 5.1.

4.2 Experimental Results

This section presents our experimental results related to watermark generation and detection.

Watermark Generation The VM image templates of Ubuntu and CentOS were copied and a total of 10 and 100 watermark fragment files were embedded in the two VM images. Each experiment was repeated 10 times and the average turnaround times were calculated.

Table 3. Turnaround times for watermark generation.

	Av. Generation Time (sec)		**Av. Copying**
	10 Files	**100 Files**	**Time (sec)**
Ubuntu Linux 8.04	13.121	24.001	6.903
CentOS 5.1	168.382	177.270	142.284

Table 3 presents the turnaround times for copying VM images and generating watermarks. The experiments used previously-created directory codes of template VM images because the codes are generated only the first time that the template VM images are used. The results indicate that the time for watermark generation excluding the time for

Table 4. Generated watermark fragment files.

File Names
`/usr/share/zoneinfo/Pacific/kQGidyI`
`/etc/console-tools/v2uQizvontwL8`
`/lib/modules/2.6.24-16-virtual/kernel/sound/usb/snd-usb-INc.ko`
`/usr/lib/klibc/bin/hso.shared`
`/usr/lib/perl/5.8.8/IO/Socket/vS9jkE.pm`
`/usr/lib/python2.5/wsgiref/hG272nT.pyc`
`/usr/share/debconf/jAGX7mQ.sh`
`/usr/share/zoneinfo/Mexico/irQ7KA`
`/usr/share/zoneinfo/posix/Chile/5XwjA`

copying is essentially independent of the size of the template VM image. The turnaround times drop when faster storage devices are employed.

Table 4 lists the generated watermark fragment files. The file names `snd-usb-INc.ko` and `hso.shared` are relatively difficult to distinguish from the pre-existing files in the target directories. In contrast, file names such as `kQGidyI` are easily distinguishable. In general, when there are many files in the target directory whose names have common extensions and long common substrings, the generated file name(s) tend to be indistinguishable. For example, the file `snd-usb-INc.ko` belongs to */lib/modules/2.6.24-16-virtual/kernel/sound/usb/* whose pre-existing files are `snd-usb-audio.ko` and `snd-usb-lib.ko`. Meanwhile, if only a few file names have common substrings/extensions in the target directory, the generated file name(s) tend to be contrived.

Watermark Detection Watermark detection experiments were conducted to identify 10 and 100 watermark fragment files from the watermarked Ubuntu 8.04 and CentOS 5.1 VM images.

Table 5. Turnaround times for watermark detection.

	Av. Detection Time (sec)	
	10 Files	**100 Files**
Ubuntu Linux 8.04	7.524	7.549
CentOS 5.1	7.526	7.573

Table 5 presents the average turnaround times for watermark detection based on ten repetitions. The results indicate that the time taken to detect the watermark is almost independent of the size of the watermarked VM image and the number of watermark fragment files.

5. Conclusions

The digital watermarking scheme for VM images is independent of file systems, file formats, operating systems, kernel versions and hardware. Also, experiments demonstrate that watermark generation and detection are both fast and effective.

Our future work will focus on enhancing the tamper-resistant characteristics of the watermarking scheme. Several attacks could be devised to remove or modify VM image watermarks. For example, watermarks could be detected using a machine learning algorithm such text clustering [9] to discriminate embedded files from pre-existing files based on file name string features. Another attack could use information from elsewhere in the VM such as the i-node numbers of files. Files installed at a given time are assigned successive i-node numbers; a file with an i-node number that is out of sequence could correspond to an embedded file. Other problems to be investigated include having the contents of embedded files resemble those of pre-existing files, and augmenting the scheme with other data hiding approaches to enhance tamper resistance. Finally, our research will investigate the application of the watermarking scheme to other digital content such as tar and zip archives.

References

[1] Amazon Web Services, Amazon Elastic Compute Cloud, Seattle, Washington (aws.amazon.com/ec2).

[2] Amazon Web Services, Amazon Simple Storage Service, Seattle, Washington (s3.amazonaws.com).

[3] Amazon Web Services, ec2-download-bundle, Seattle, Washington (docs.amazonwebservices.com/AmazonEC2/dg/2006-10-01/CLTRG-ami-download-bundle.html).

[4] L. Blum, M. Blum and M. Shub, Comparison of two pseudorandom number generators, in *Advances in Cryptology: Proceedings of Crypto 1982*, D. Chaum, R. Rivest and A. Sherman (Eds.), Plenum, New York, pp. 61–78, 1982.

[5] K. Eckstein and M. Jahnke, Data hiding in journaling file systems, *Proceedings of the Fifth Annual Digital Forensic Research Workshop*, 2005.

[6] A. McDonald and M. Kuhn, StegFS: A steganographic file system for Linux, *Proceedings of the Third International Workshop on Information Hiding*, pp. 463–477, 2000.

[7] F. Perez-Gonzalez and J. Hernandez, A tutorial on digital watermarking, *Proceedings of the Thirty-Third IEEE International Carnahan Conference on Security Technology*, pp. 286–292, 1999.

[8] N. Quynh and Y. Takefuji, A novel approach for a file-system integrity monitor tool for a Xen virtual machine, *Proceedings of the Second ACM Symposium on Information, Computer and Communications Security*, pp. 194–202, 2007.

[9] T. Segaran, *Programming Collective Intelligence: Building Smart Web 2.0 Applications*, O'Reilly, Sebastopol, California, 2007.

Chapter 19

A VISUALIZATION SYSTEM FOR ANALYZING INFORMATION LEAKAGE

Yuki Nakayama, Seiji Shibaguchi and Kenichi Okada

Abstract Information leakage is a growing public concern. This paper describes a visualization system for tracing leaks involving confidential information. In particular, the system enables administrators to determine which hosts have confidential documents and the means by which confidential information is transmitted, received and duplicated. The visualization system is scalable to large organizations and can track various means of information propagation in a seamless manner. Also, it helps prevent information leaks, analyze transmission routes and present forensic evidence.

Keywords: Information leakage, visualization, data tracing

1. Introduction

Information leakage has become a serious problem. A recent survey of more than 800 CIOs reported that organizations lost an average of $4.6 million due to the leakage of intellectual property [7]. It is imperative to protect intellectual property and sensitive information by devising countermeasures against information leakage.

Countermeasures against information leakage can be broadly classified as before-the-fact or after-the-fact measures. Before-the-fact countermeasures aim to prevent leakage (e.g., prohibiting the use of USB drives, printing confidential documents or sending confidential data by email). After-the-fact measures involve incident response and digital forensic investigations [12]. Digital forensics is the use of scientifically derived and proven technical methods and tools for the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence for the purpose of facilitating or furthering the reconstruction of events [13].

Research conducted in Japan [3, 9] indicates that information leakage by insiders is about 1% of the total and that human error, such as employees losing their laptop computers, is the main cause of information leakage. Similar results have been observed in the United States [2], where human error contributes to as much as 35.2% of the information leakage that occurs in the private sector. Consequently, it is important to focus on preventing leakage due to human error.

Our visualization system contributes to both before-the-fact and after-the-fact countermeasures. Specifically, it traces the pathways of confidential information and supports the visualization of the routes. It allows administrators to know which users have confidential documents, enabling them to remove the documents or to prohibit the users from using laptops outside the enterprise. Therefore, leakage due to human error, such as the loss of a laptop, can be prevented. Furthermore, the system contributes to digital forensic investigations by enabling administrators to rapidly analyze the cause of a leakage and presenting forensic evidence using an intuitive interface.

2. Related Work

This section discusses some of the existing tools for combating information leakage and highlights their deficiencies.

Vontu Data Loss Prevention [11] provides proactive countermeasures against information leakage. Also, it implements systematic security controls such as restricting the use of USB drives. However, these controls hinder routine work and lower productivity. Also, they can only prevent information leakage that occurs in a predictable manner. On the other hand, humans can analyze problems from various perspectives and make systematic judgments about the risk of leakage at any given time. Our visualization system assists humans in understanding rapidly changing situations that may involve information leakage.

SKYSEA Client View [10] and InfoCage [4, 5, 8] support after-the-fact countermeasures. These tools monitor the operation of hosts and write the results into text-based log files. They also analyze the log files to ascertain the cause of information leakage. However, they do not attempt to streamline analytic work or simplify evidence for presentation in court. Consequently, administrators must devote considerable amount of time and effort to analyzing large amounts of log data. Eliminating this step is important to simplifying analytic work. Our visualization system facilitates the presentation of evidence related to information leakage in a clear, concise and simple manner.

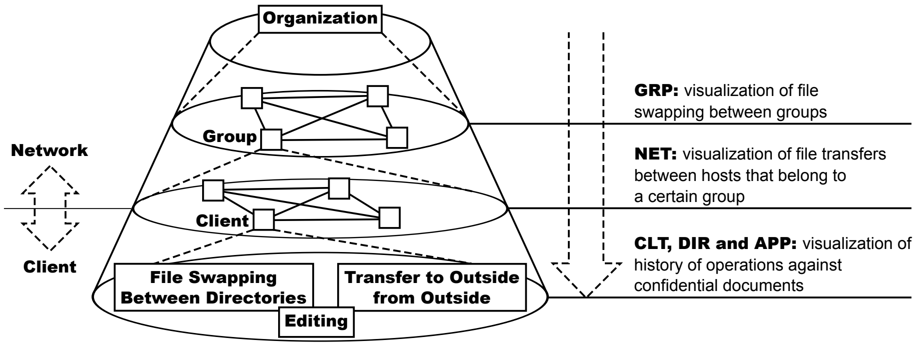

Organization
Group
Client
Network
Client
File Swapping Between Directories
Transfer to Outside from Outside
Editing
GRP: visualization of file swapping between groups
NET: visualization of file transfers between hosts that belong to a certain group
CLT, DIR and APP: visualization of history of operations against confidential documents

Figure 1. Scalable and seamless visualization.

3. Visualization System

Our visualization system monitors the transmission of confidential information via pathways such as email, removable media and applications. It enables administrators to identify the locations of confidential documents. If necessary, they can then implement security measures such as ordering employees to remove the sensitive documents or prohibiting the removal of specific computers from the enterprise.

The visualization system also contributes to the rapid analysis and clear presentation of forensic evidence in the event of an information leakage. Moreover, damage to the organization is minimized due to the rapid response.

Two key requirements related to visualizing information transmission pathways are:

- **Scalable Visualization:** A visualization system must be flexible with regard to changes in the number of hosts and the number of confidential documents that appear on the watch list.

- **Seamless Visualization:** Information can be propagated by various means – file swapping via email or peer-to-peer networks, transporting data on portable devices such as USB flash drives, moving or copying files to a computer, and editing or duplicating files using applications. A visualization system should achieve broad coverage of these diverse means of data transmission and integrate the means seamlessly.

Our system addresses these requirements using five different visualization methods (Figure 1). The five methods are group-based (GRP), network-based (NET), client-based (CLT), directory-based (DIR) and application-based (APP) methods. GRP and NET observe file swap-

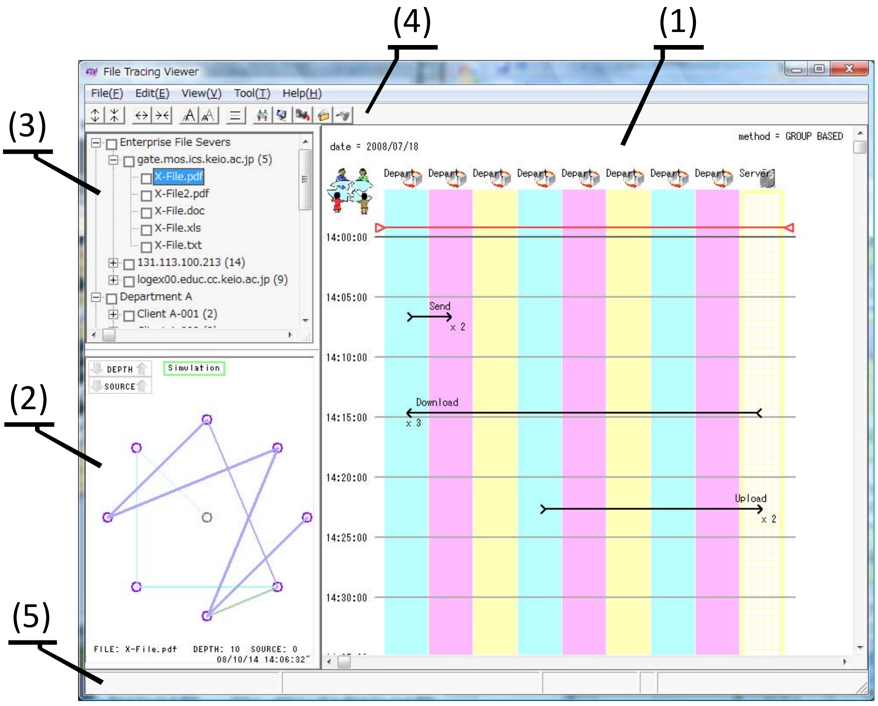


Figure 2. CROWS Up Viewer.

ping through networks. CLT, DIR and APP monitor computer use by employees. GRP visualizes file swapping between groups comprising a given number of hosts. NET displays file transfers between hosts that belong to targeted groups. CLT monitors confidential documents received and dispatched by a particular client. DIR shows the transfer of confidential documents between the directories of a single client. APP monitors applications when a user has confidential documents open.

4. CROWS Up Viewer

We have implemented a prototype called the "CROWS Up Viewer" (CROWS: Catch Reveal by Observing and WitneSsing), which meets the design goals described in Section 3. The system is implemented in C++ and runs under Windows XP/Vista.

Figure 2 shows the CROWS Up Viewer. The main panel (marked (1)) presents the primary interface for each of the five visualization methods. The sub-panel (2) shows the sub-methods that assist with analytic work. The tree view (3) shows the documents possessed by each host. The

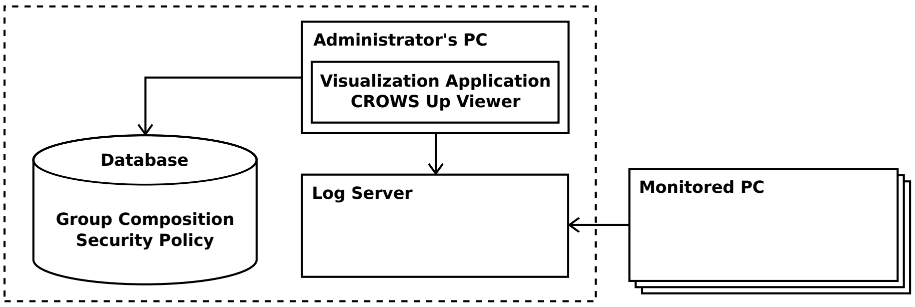


Figure 3. System architecture.

toolbar (4) provides buttons for user interaction and enables users to customize the CROWS Up Viewer interface. The status bar (5) shows details regarding the position of the on-screen cursor.

4.1 System Architecture

Figure 3 shows the visualization system architecture. A monitoring program is required to be installed on client computers in advance. This program is linked to a database, which stores a definition file for each client machine and records data. The monitoring program refers to the definition file for each client and writes results to log files, which are transmitted to a management server. Administrators use the visualization system to view the collected data.

The monitoring program uses API hooking to reveal the internal operations of the computer. This approach has been used in intrusion prevention [1], dynamic malware analysis [14] and unknown virus detection [6].

4.2 Visualization Methods

This section describes each of the five visualization methods used in our system.

GRP The group-based method (GRP) visualizes file swapping between groups comprising a given number of hosts. The top-left corner of Figure 4 shows an example of GRP visualization. The box-shaped areas represent groups and servers, and the arrows between the areas indicate file swapping; the other arrows represent file swapping in unsafe networks. Note that the time axis is set in vertical direction. For example, Figure 4 shows that Group A has sent two files to Group C at 14:07. Although details such as filenames are not visible in the figure,

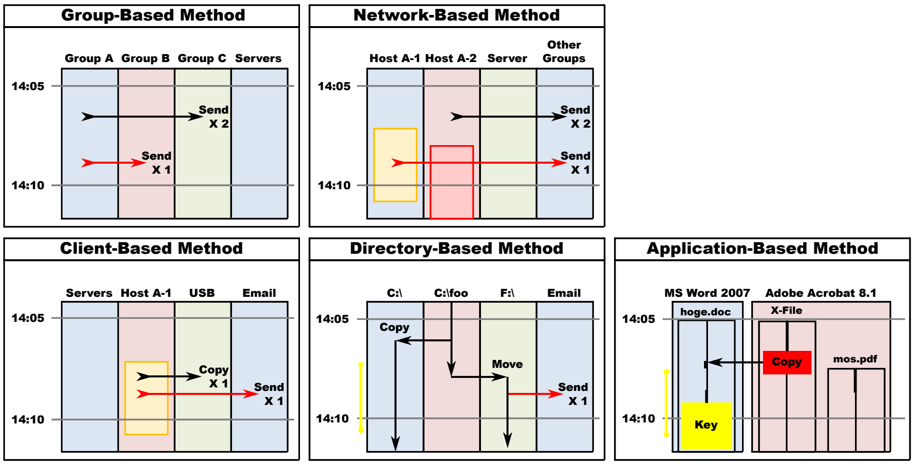


Figure 4. Visualization methods.

they can be made to appear in the toolbar by hovering the cursor or in a dialog box by clicking the mouse.

GRP provides a simulation monitor (SimMon) and a tracing monitor (TrcMon) for supporting analytical work (Figures 5 and 6). Each group is represented as a node in these monitors. The center node indicates the server group while the peripheral nodes denote client computer groups. In SimMon, the lines between the nodes represent file swapping and various colors are used to indicate operations such as file downloads, uploads and transfers (Figure 5). SimMon changes its display continually and enables administrators to run simulations that show file swapping in a dynamic workplace environment.

TrcMon shows the transmission routes of a single confidential document. The weights of the lines connecting the nodes represent the passage of time; thinner lines represent older transmissions and thicker lines represent newer transmissions. Various colors are used to highlight the branches of a document pathway. Figure 6 shows a document that has been transmitted from a server to a Group 1 user and then from Group 1 to Group 3, from Group 3 to Group 5, and eventually from Group 5 to two separate groups. It is possible to interact with TrcMon to change the time when a trace is started or to adjust its depth.

Note that GRP monitors FTP downloads and uploads, the sending of email and whether or not hosts connect to unsafe networks.

NET The network-based method (NET) displays file transfers between hosts that belong to a specified group. In Figure 4, the areas on the left represent clients, servers and other groups. The arrows between

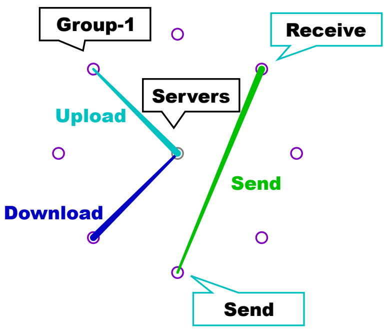


Figure 5. SimMon (GRP).

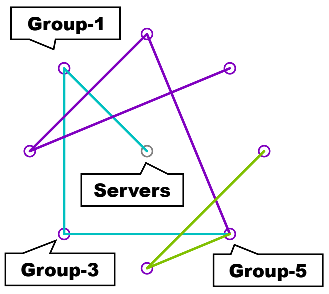


Figure 6. TrcMon (GRP).

these areas represent file swapping. Colored diagonal line areas in NET alert administrators to the status of particular clients. Hosts that are experiencing security problems (e.g., hosts that are not using antivirus software or a firewall) are shown in red. Clients that are connected to unsecured networks are shown in yellow.

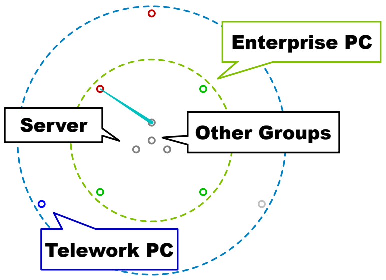


Figure 7. SimMon (NET).

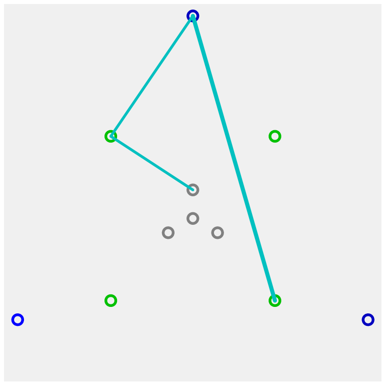


Figure 8. TrcMon (NET).

NET also provides SimMon and TrcMon (Figures 7 and 8). The node in the center represents other groups and the circularly arranged nodes represent servers and enterprise/remote clients. Nodes of various colors represent online enterprise computers (green), online remote computers (blue), offline hosts (grey) and hosts having security problems or hosts connected to unsafe networks (red).

NET monitors the downloading and uploading of files and the sending of email. Also, it monitors hosts that connect to unsafe networks or experience security problems.

CLT The client-based method (CLT) monitors confidential documents received and dispatched by a targeted client. The bottom-left corner of Figure 4 shows an example of CLT. The rectangular areas, starting from the left, represent the file servers in an enterprise, clients, removable media and email. The figure shows an example where Host A-1 has copied a confidential document to a USB flash drive at 14:08.

CLT monitors FTP downloads and uploads to enterprise servers, the sending of email, copying to or from removable media, writing to magnetic media, printing of documents and the number of confidential documents on a host.

DIR The directory-based method (DIR) shows the transfer of confidential documents between the directories of a single client. Figure 4 shows an example of DIR, where a confidential document was copied from `C:\foo\` to `C:\` at 14:06.

DIR monitors the copy, move, remove/recover, open/close, save as, print, upload, send, compress/decompress, convert and split/combine operations with respect to confidential files.

APP The application-based method (APP) monitors all applications when a user has confidential documents open. Figure 4 shows an example of its operation. Each application is drawn as an area on the display in which APP displays the confidential documents opened by the application. The heavy lines represent active windows. In the example, the user has copied text data from `X-File` to `hoge.doc` at 14:07.

APP also provides DspMon (display monitor) as shown in Figure 9. DspMon duplicates a computer screen at a given time and provides an intuitive representation. A framed rectangle denotes an active window and underlined filenames indicate that the files are confidential.

APP monitors the copying/pasting of text and bitmap data; the position/size of application windows; the z-index of windows; keystrokes; the opening/closing of documents; and save/save as and print commands.

4.3 Analytic Workflow

This section describes an example analytic workflow conducted using the CROWS Up Viewer. In the example, we assume that an administrator has learned that a file has been leaked and attempts to analyze the cause of the leakage.

First, the administrator examines the tree view in the upper left portion of the CROWS Up Viewer and selects the file to be traced. At this point, the CROWS Up Viewer removes all the other files from view.

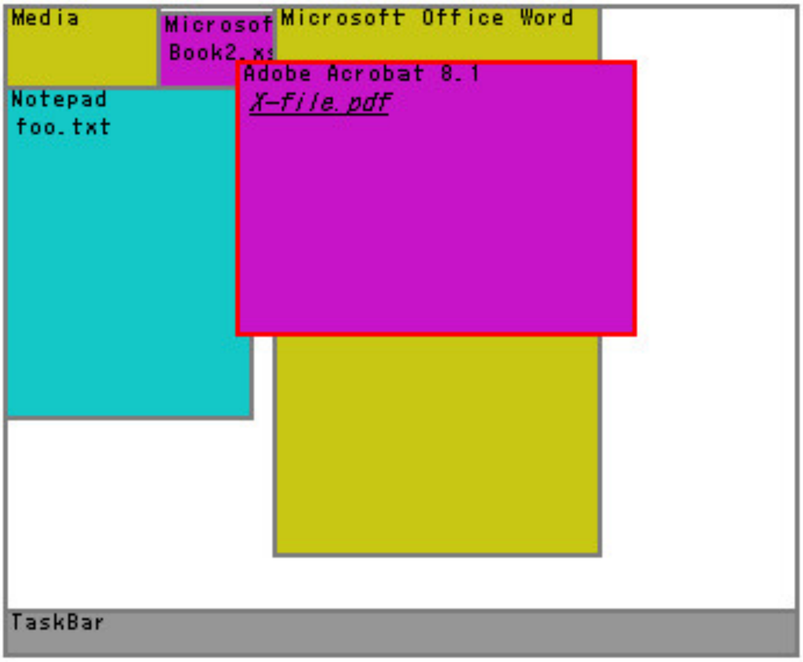


Figure 9. DspMon.

The administrator then employs the five visualization methods in sequence (Figure 10). First, the groups that received the file are analyzed using GRP, after which, the administrator applies NET to the groups to identify the hosts that received the leaked file. It is important to note that the groups that transfer a file via an unsafe network receive a higher priority. Using NET, the administrator first observes the colored diagonal areas that identify clients with security problems (red) and clients connected to unsafe networks (yellow). Thus, the administrator can check if viruses, worms or other malicious software caused the leakage or if the file was stolen by data sniffing.

Next, the administrator uses CLT, DIR and APP to investigate the hosts flagged by NET. CLT provides information about incoming and outgoing transfers of the file; DIR provides information about the use of the file; APP shows the operations that the user performed using various applications.

The CROWS Up Viewer enables the administrator to analyze the situation intuitively even if the events are complex. As a result, the cause of the file leakage can be determined promptly and appropriate mitigation actions can be taken.

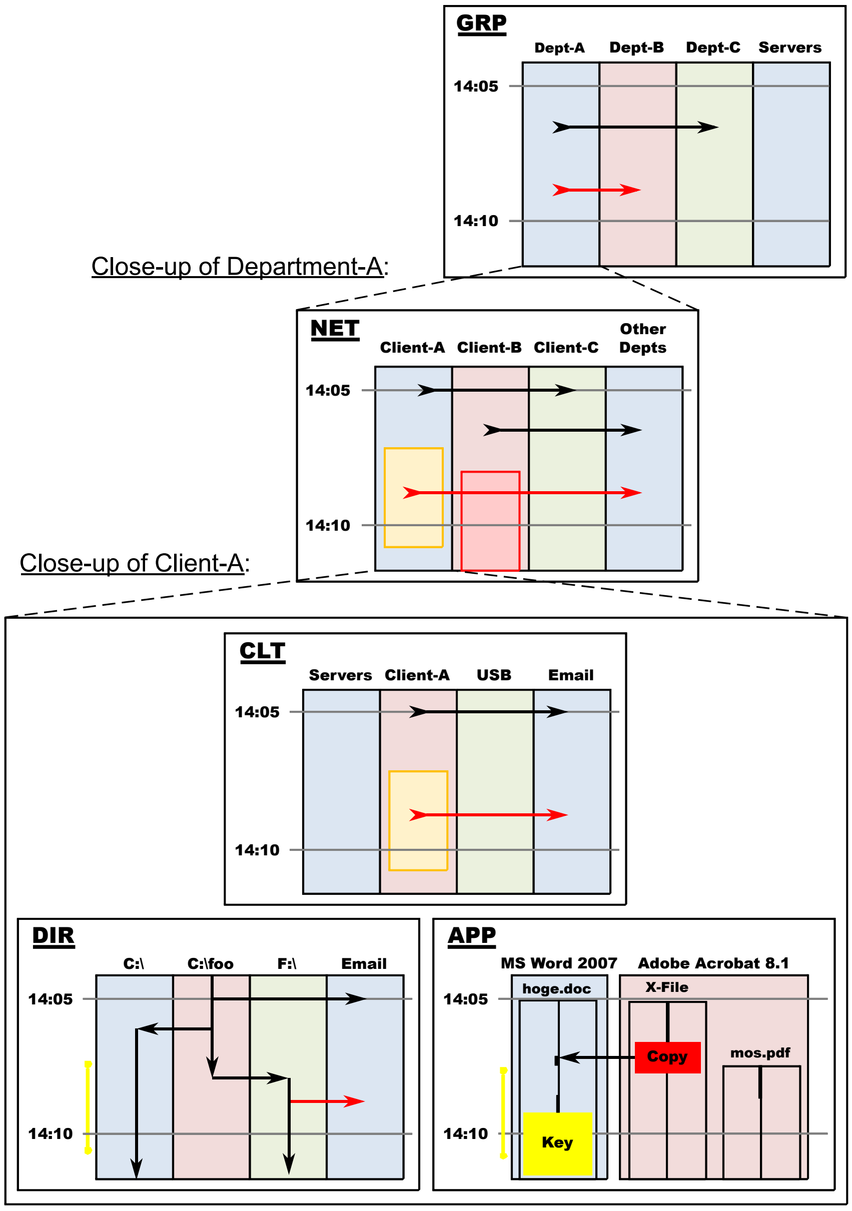


Figure 10. Analytic workflow.

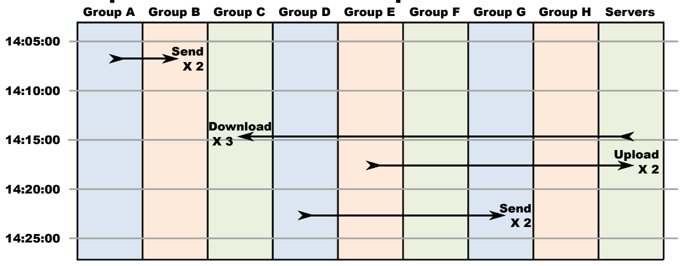
5. Experimental Results

We conducted an experiment to verify the usability of the visualization system. The subjects of the experiment were twenty university students who were pursuing degrees in information engineering. The

File-swappings between groups

Time	Operation	Source	Destination	# Files
14:07:30	Send	Group A	Group B	2
14:15:23	Download	Server	Group C	3
14:18:56	Upload	Group E	Server	2
14:07:30	Send	Group D	Group G	2

Question:
Group A sends 2 files to Group B at 14:07:30.



Question:
Group E uploaded 2 files to a server at 14:18.

Figure 11. Sample evaluation questions.

subjects performed simple analytic tasks using text logs and visualized logs, and answered 26 true/false questions about the tasks – thirteen questions regarding the text logs and thirteen regarding the visualized logs (Figure 11). We evaluated three characteristics of the visualization system: (i) accuracy; (ii) speed; and (iii) ease of understanding based on the percentage of correct answers to the survey questions and the time required to answer the questions.

The subjects were divided into two groups to address the possibility that they might become accustomed to the analytic work, which could affect the comparison between the text log and visualized log results. Subjects in Group 1 analyzed the text logs first, followed by the visualized logs. Subjects in Group 2 answered questions about the visualized logs first, followed by questions about the text logs. Each group comprised ten subjects.

Table 1 shows the results of the experiment. P_{txt} is the percentage of correct answers for the text logs; P_{viz} is the percentage of correct answers for the visualized logs; FS is fractional reduction in the time required $((T_{txt} - T_{viz})/T_{txt})$, where T_{txt} is the time required for analyzing the text logs and T_{viz} is the time required for analyzing the visualized logs.

An F-test and a t-test were conducted between: (i) P_{txt} of Group 1 and P_{txt} of Group 2; (ii) P_{viz} of Group 1 and P_{viz} of Group 2; and

Table 1. Experimental results.

Group 1			Group 2		
P_{txt}	P_{viz}	FS	P_{txt}	P_{viz}	FS
0.92	1.00	+0.43	0.92	1.00	+0.60
0.85	1.00	+0.37	0.77	1.00	+0.25
0.62	1.00	+0.16	0.92	1.00	+0.20
1.00	1.00	+0.45	0.92	0.92	+0.26
0.85	0.92	+0.62	0.92	1.00	+0.34
0.85	1.00	+0.20	1.00	1.00	+0.40
0.85	1.00	+0.17	1.00	1.00	+0.31
0.62	0.92	+0.33	0.85	0.92	+0.39
1.00	1.00	+0.30	1.00	1.00	+0.32
0.92	1.00	+0.19	0.92	1.00	+0.28

(iii) FS of Group 1 and FS of Group 2. No significant differences were observed for a significance level of 5%. Thus, we can conclude that no significant difference exists between Group 1 and Group 2, i.e., the tests can be regarded as having been conducted under the same conditions.

The means and standard deviations (upon combining Groups 1 and 2) are:

$$P_{txt} = 88 \pm 11\% \qquad P_{viz} = 98 \pm 3\% \qquad FS = +34 \pm 13\%$$

The percentage of correct answers related to the visualized logs were higher than those for the text logs, and no significant difference exists at a level of 1% ($p < .01$). The fractional reduction in the time required indicates that the visualized logs facilitate rapid analysis. Therefore, the visualization system supports accurate and rapid analysis.

After the experiment, we also polled the test subjects about the ease of understanding of the visualized logs versus the text logs. All the subjects felt that the visualized logs were easier to understand than the text logs.

6. Conclusions

Information leakage is a serious problem and it is imperative that organizations employ effective countermeasures to protect confidential information. Our visualization system efficiently traces the flow of confidential information and helps identify potential information leaks, enabling administrators to assess the risk and implement mitigation strategies. The system also supports forensic investigations of information leaks and assists with the collection and presentation of evidence. Experimental results indicate that the visualization system supports human understanding and facilitates the rapid and accurate analysis of information leaks.

References

[1] R. Battistoni, E. Gabrielli and L. Mancini, A host intrusion prevention system for Windows operating systems, *Proceedings of the Ninth European Symposium on Research on Computer Security*, pp. 352–368, 2004.

[2] Identity Theft Resource Center, 2008 data breach totals soar, Press Release, San Diego, California (www.idtheftcenter.org/artman2/publish/m_press/2008_Data_Breach_Totals_Soar.shtml), 2009.

[3] Information-Technology Promotion Agency, Countermeasures Against Information Leakage: Seven Rules for People Working in Business Enterprises, Tokyo, Japan (www.ipa.go.jp/security/english/virus/antivirus/pdf/Leakage_measures_eng.pdf), 2006.

[4] M. Kawakita, K. Yanoo, M. Hosokawa, H. Terasaki, S. Aoki and T. Usuba, InfoCage – Information leakage protection software, *NEC Journal of Advanced Technology*, vol. 2(1), pp. 40–46, 2005.

[5] K. Kida, H. Sakamoto, H. Shimazu and H. Terumi, InfoCage: A development and evaluation of confidential file lifetime monitoring technology by analyzing events from file systems and GUIs, *Proceedings of the Second International Workshop on Security*, pp. 246–261, 2007.

[6] R. Koike, N. Nakaya and Y. Koui, Development of a USB flash memory for detecting computer viruses, *Information Processing Society of Japan Journal*, vol. 48(4), pp. 1595–1605, 2007.

[7] McAfee, Unsecured Economies: Protecting Vital Information, Santa Clara, California, 2009.

[8] NEC Corporation, No. 1 market share for domestic quarantine tools for three consecutive years, Press Release, Tokyo, Japan (www.nec.co.jp/press/ja/0808/2701.html), 2008.

[9] Security Incident Investigation Working Group, Survey Report of Information Security Incidents 2007, Version 1.0, NPO Japan Network Security Association, Tokyo, Japan (www.jnsa.org/result/2007/pol/incident/2007incidentsurvey_e_v1.0.pdf), 2008.

[10] Sky Corporation, SKYSEA Client View, Osaka, Japan (www.skyseaclientview.net).

[11] Symantec Corporation, Data loss prevention: Products and services, Mountain View, California (www.symantec.com/business/theme.jsp?themeid=vontu).

[12] S. Tsujii and E. Hagiwara (Eds.), *Encyclopedia of Digital Forensics*, Nikkagiren Press, Tokyo, Japan, 2008.

[13] S. Willassen and S. Mjolsnes, Digital forensics research, *Telektronikk*, vol. 2005(1), pp. 92–97, 2005.

[14] C. Willems, T. Holz and F. Freiling, Toward automated dynamic malware analysis using CWSandbox, *IEEE Security and Privacy*, vol. 5(2), pp. 32–39, 2007.

VI

FORENSIC TOOLS

Chapter 20

FORENSIC ANALYSIS OF POPULAR CHINESE INTERNET APPLICATIONS

Ying Yang, Kam-Pui Chow, Lucas Hui, Chunxiao Wang, Lijuan Chen, Zhenya Chen and Jenny Chen

Abstract When the Digital Evidence Search Kit (DESK) was first used in Mainland China, it was found to be inadequate because it did not support criminal investigations involving popular Internet applications such as QQ, MSN and Foxmail. This paper discusses the enhancements made to DESK to conduct forensic analyses of QQ, MSN and Foxmail.

Keywords: Forensic analysis, Internet applications, QQ, MSN, Foxmail

1. Introduction

The Digital Evidence Search Kit (DESK) is a digital forensic tool developed by the Center for Information Security and Cryptography at the University of Hong Kong [1]. DESK was introduced to Mainland China in 2007 and is now being used on a trial basis by several Chinese public security departments. During the performance review phase, it was discovered that DESK was inadequate for investigations in China because it did not support forensic analyses of some Internet applications that are popular in China, but rarely used elsewhere in the world. These applications, which include QQ, MSN and Foxmail, have special data formats and often encrypt important data.

QQ [4] is an instant messaging (IM) application with Chinese characters developed by Tencent Holdings; MSN is a similar application developed by Microsoft. According to the 2007-2008 China Internet Survey, QQ has an overwhelming market share in China. QQ is followed by MSN, which is the most popular instant messaging tool among office workers. Other instant messaging tools have a miniscule market share.

QQ's popularity stems from the fact that it is designed to accommodate Chinese Internet communication habits, including the need to communicate with friends and strangers. QQ is positioned as a comprehensive platform for entertainment, Internet chat and communications. It continuously incorporates peripheral functions that are attractive to children and young adults, causing the numbers of new QQ accounts and new QQ users to increase dramatically. According to an iResearch report, the QQ IM application had 132,740,000 active users in China as of June 2009. Indeed, almost every netizen in China has a QQ account.

Another popular system is Foxmail [8], a client-side email software also developed by Tencent Holdings. Foxmail functions are similar to Microsoft Outlook, but with specialized Chinese character support, which has contributed to its widespread use in China.

Meanwhile, the use of IM and email by criminal entities is growing in China. In particular, criminals use QQ, MSN and Foxmail to disseminate obscene images or links to pornographic websites, to divulge national secrets, and even to discuss, plan and coordinate criminal operations. This paper discusses the enhancements made to DESK to support forensic investigations involving QQ, MSN and Foxmail.

2. QQ Forensic Analysis

This section presents key details of the QQ IM application and outlines a forensic analysis methodology.

2.1 Overview

Every new user must register with the QQ service before using the IM application. Upon successful registration, the user is assigned a QQ number (i.e., a QQ account). The user may then log into the QQ service using the QQ number and start a session. The QQ processing can be summarized as follows:

- When a user logs into the QQ service, the QQ client obtains the latest friends list from the QQ server and establishes a peer-to-peer (P2P) connection between the user and each friend on the list.
- The user and his/her friends communicate using UDP.
- If a P2P connection cannot be established (e.g., due to a network problem), messages between the two friends are transmitted through the QQ server. The server stores all messages that the sender has sent but the receiver has not yet received. The stored messages are passed to the receiver when he/she next logs into the QQ service.

Table 1. QQ packet structure.

Bytes	Content
0	Start of packet (`0x02`)
1 to 2	QQ version number (expressed using network byte order)
3 to 4	Command number (expressed using network byte order)
5 to 6	Sending serial number (receiver should check the number)
7 to n	QQ data (possibly encrypted)
n+1	End of packet (`0x03`)

QQ messages are sent using UDP. Each UDP packet has no more than 64K bytes. Table 1 presents the packet structure.

Table 2. QQ files.

QQ File	Function
`QQApplication.dll`	Friends panel display program
`LoginUinList.dat`	Login history file
`QQZip.dll`	Compression and decompression utilities
`QQMainFrame.dll`	QQ main panel
`Newface`	Directory containing all portrait files
`QQ.exe`	QQ main executable
`QQFileTransfer.dll`	QQ file transfer utility
`QQHook.dll`	QQ keyboard monitor program
`QQPlugin.dll`	QQ friends searching utility
`QQRes.dll`	QQ resource handling function

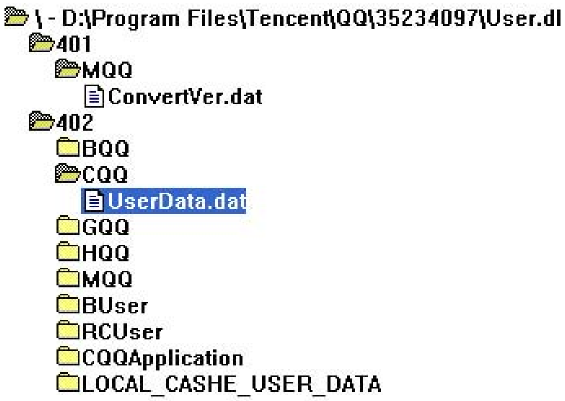
2.2 Principal Files

In order to perform QQ forensics, it is important to understand the file organization. Table 2 lists the files present in a QQ directory after the successful installation of the application.

User account information is stored in several files in a directory named after the user's QQ number (e.g., 12345678). The principal files are:

- `MsgEx.db`: This file is created after the user registers with and logs into the QQ service. The file stores all chat records using structured storage [3]. Local history data, which includes chat records and logs, are encrypted using TEA [7] and stored in `MsgEx.db`.

- `ewh.db`: This file stores the MD5 hash [5] of the user's password. When a user attempts to log into the QQ service, the QQ client verifies the submitted password with the password stored in the



```
\ - D:\Program Files\Tencent\QQ\35234097\User.dl
    401
        MQQ
            ConvertVer.dat
    402
        BQQ
        CQQ
            UserData.dat
        GQQ
        HQQ
        MQQ
        BUser
        RCUser
        CQQApplication
        LOCAL_CASHE_USER_DATA
```

Figure 1. `User.db` data structure.

`MsgEx.db` file. The QQ server then performs a second validation before the user can successfully log in. This involves hashing the user-supplied password and comparing the value with the password hash saved in `ewh.db`.

- `Notes.db`: This file stores the QQ memorandum.
- `User.db`: This file stores the friend records. It uses the same structured storage format and TEA encryption as `MsgEx.db`.
- `QQAVFile`: This directory stores all QQ image files.
- `CustomFace`: This directory stores all self-defining expressions.
- `CustomFaceRecv`: This directory stores all received self-defining expressions.
- `ShareInfo.db`: This file stores the configuration information of the shared directory.

2.3 Key Technologies

Structured Storage Structured storage was developed by Microsoft for storing hierarchical data in the Windows operating system [3]. It improves disk space efficiency and simplifies software distribution by gathering all the data files into one file. In the structured storage paradigm, storage can contain other storage, just like a directory can contain subdirectories. The `MsgEx.db` and `User.db` files store data using structured storage. Figure 1 presents a sample `User.db` structured storage file.

Table 3. Registry information.

Registry Field	**Description**
HKEY_LOCAL_MACHINE\SOFTWARE\Tencent\QQ	
Install=c:\Program Files\Tencent\qq	QQ installation path
version=1413.192	QQ version number

MD5 Hashing MD5 [5] is a popular cryptographic hash algorithm that converts a variable-length message into a 128-bit hash. The input message is broken up into chunks of 512-bit blocks. The output consists of four sub-groups of 32 bits, which are cascaded to form the 128-bit hash value. QQ uses MD5 to generate the encryption key from the user's QQ number.

TEA Encryption Tiny Encryption Algorithm (TEA) [7] implements a block cipher that uses a 128-bit key and operates on 64-bit blocks. It has a Feistel structure with suggested 64 rounds, typically implemented in pairs called "cycles." It has a very simple key schedule, mixing all the key material in exactly the same way for each cycle. Different multiples of a magic constant are used to prevent simple attacks based on the symmetry of the rounds. The magic constant 2654435769 (`9E3779B916`) is computed as $\left\lfloor \frac{2^{32}}{\phi} \right\rfloor$ where ϕ is the golden ratio. Although 64 rounds are suggested for security reasons, QQ uses only sixteen rounds of TEA to encrypt the `MsgEx.db` and `User.db` files.

2.4 Forensic Analysis

Analyzing the Registry Table 3 lists the Windows registry information maintained about the QQ application after a successful installation. The QQ installation information may be extracted by exploring the Windows registry.

Decrypting Data One of the important tasks in QQ forensics is to extract the encrypted chat records from the `MsgEx.db` file. Figure 2 presents the structured storage scheme used by `MsgEx.db` to store data. `C2CMsg` stores chat record messages, `SysMsg` stores system messages and `GroupMsg` stores group messages. The message contents themselves are stored in `Data.msj` files. Peer-to-peer messages are stored in `Data.msj` files under the QQ number directory within the `C2CMsg` folder and are indexed by the `Index.msj` file. Group messages are stored in `Data.msj`

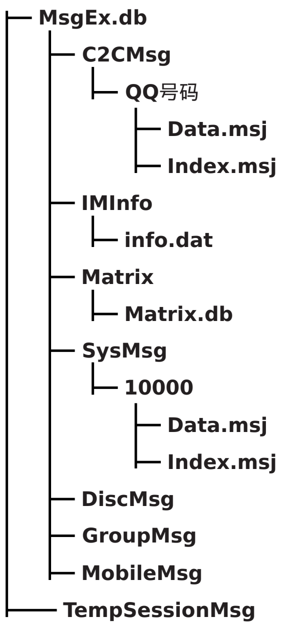


Figure 2. `MsgEx.db` data structure.

files under the directory "SysMsg\10000" and are indexed by `Index.msj` in the same directory. The file is encrypted using TEA.

The primary task in QQ forensics is to extract the decrypted data from `C2CMsg`, `SysMsg` and `GroupMsg`. The following steps are involved in decryption and extraction:

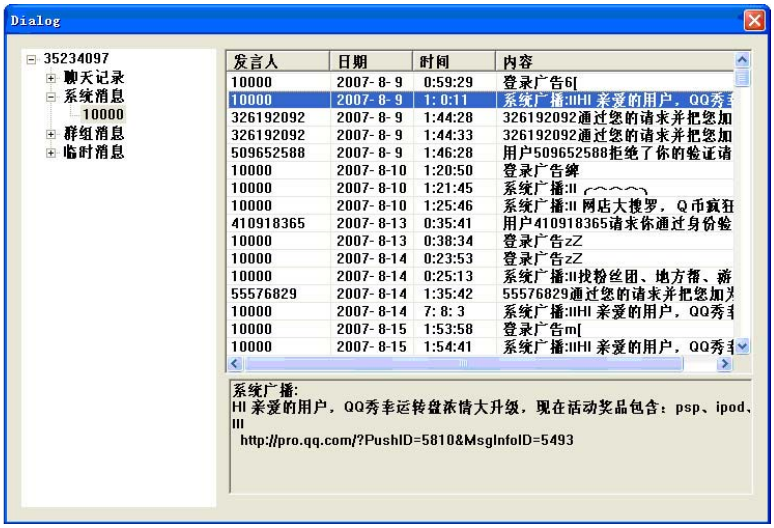
- Obtain the directory and the QQ number and transform the QQ number to the MD5 key using the MD5 algorithm.

- Obtain data from `Matrix.db` for `C2CMsg`, `SysMsg` and `GroupMsg`. Note that QQ often applies padding and permutation. Decrypt the data using sixteen rounds of TEA with the MD5 key.

- Translate the decrypted chat records and friends list into Chinese and display the chat record messages as shown in Figure 3.

3. MSN Forensic Analysis

This section presents key details of the MSN application and outlines a forensic analysis methodology.



发言人	日期	时间	内容
10000	2007- 8- 9	0:59:29	登录广告6[
10000	2007- 8- 9	1: 0:11	系统广播:IIHI 亲爱的用户，QQ秀
326192092	2007- 8- 9	1:44:28	326192092通过您的请求并把您加
326192092	2007- 8- 9	1:44:33	326192092通过您的请求并把您加
509652588	2007- 8- 9	1:46:28	用户509652588拒绝了你的验证请
10000	2007- 8-10	1:20:50	登录广告缔
10000	2007- 8-10	1:21:45	系统广播:II ╭⌒⌒⌒╮
10000	2007- 8-10	1:25:46	系统广播:II 网店大搜罗，Q币疯狂
410918365	2007- 8-13	0:35:41	用户410918365请求你通过身份验
10000	2007- 8-13	0:38:34	登录广告zZ
10000	2007- 8-14	0:23:53	登录广告zZ
10000	2007- 8-14	0:25:13	系统广播:II找粉丝团、地方帮、游
55576829	2007- 8-14	1:35:42	55576829通过您的请求并把您加
10000	2007- 8-14	7: 8: 3	系统广播:IIHI 亲爱的用户，QQ秀
10000	2007- 8-15	1:53:58	登录广告m[
10000	2007- 8-15	1:54:41	系统广播:IIHI 亲爱的用户，QQ秀

系统广播:
HI 亲爱的用户，QQ秀幸运转盘浓情大升级，现在活动奖品包含：psp、ipod、
III
http://pro.qq.com/?PushID=5810&MsgInfoID=5493

Figure 3. QQ chat records.

3.1 Overview

After a user installs MSN and executes the software, several digital traces remain, these include the MSN system configuration, friend's messages and communication messages.

The MSN installation directory is recorded in the registry field:

HKEY_LOCALMACHINE\SOFTWARE\Microsoft
\MSNMessenger\InstallationDirectory.

Each MSN user account has its own configuration settings, which are recorded in the registry field:

HKEY_CURRENTUSER\Software\Microsoft
\MSNMessenger\PerPassportSettings\⋆

where "⋆" denotes PassID, which is generated from the user name.

The default path for storing MSN chat records is:

%SysDisk%\Documents and Settings\(Windows login user)
\My Documents\My Received Files\(emailName+PassID)
\history.

The user may change the default path, which is then stored in the registry field:

HKEY_CURRENTUSER\Software\Microsoft
\MSNMessenger\PerPassportSettings\PassID
\MessageLogPath.

MSN chat records are stored in files using the XML format. The filename is of the form `user-nick-name+account-number+.xml` [6], where

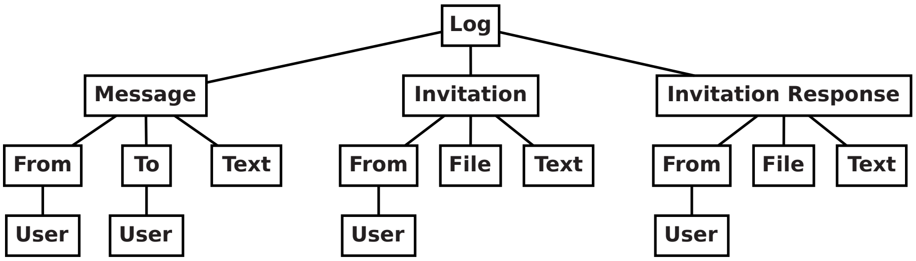
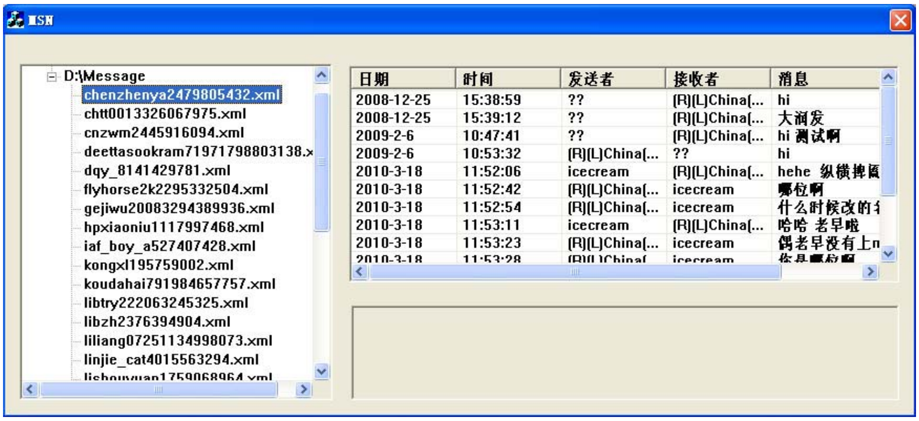


Figure 4. XML file tree structure.

`user-nick-name` is the nickname of conversation counterpart. Each file stores the chat records of the user and file information (name and path) that the user has received. The XML file uses a tree structure to store the chat records (Figure 4).



D:\Message
- chenzhenya2479805432.xml
- chtt0013326067975.xml
- cnzwm2445916094.xml
- deettasookram71971798803138.x
- dqy_8141429781.xml
- flyhorse2k2295332504.xml
- gejiwu20083294389936.xml
- hpxiaoniu1117997468.xml
- iaf_boy_a527407428.xml
- kongxl195759002.xml
- koudahai791984657757.xml
- libtry222063245325.xml
- libzh2376394904.xml
- liliang07251134998073.xml
- linjie_cat4015563294.xml
- lishouyuan1759068964.xml

日期	时间	发送者	接收者	消息
2008-12-25	15:38:59	??	(R)(L)China(...	hi
2008-12-25	15:39:12	??	(R)(L)China(...	大润发
2009-2-6	10:47:41	??	(R)(L)China(...	hi 测试啊
2009-2-6	10:53:32	(R)(L)China(...	??	hi
2010-3-18	11:52:06	icecream	(R)(L)China(...	hehe 纵横捭阖
2010-3-18	11:52:42	(R)(L)China(...	icecream	哪位啊
2010-3-18	11:52:54	(R)(L)China(...	icecream	什么时候改的
2010-3-18	11:53:11	icecream	(R)(L)China(...	哈哈 老早啦
2010-3-18	11:53:23	(R)(L)China(...	icecream	偶老早没有上n
2010-3-18	11:53:28	(R)(L)China(	icecream	你是哪位啊

Figure 5. MSN chat records.

3.2 Forensic Analysis

The chat record files may be processed by a standard XML parser (e.g., Microsoft's XML parser) to extract the necessary information. Figure 5 shows example MSN chat records that are obtained using this method.

4. Foxmail Forensic Analysis

This section presents key details of the Foxmail application and outlines a forensic analysis methodology.

4.1 Overview

Foxmail emails are stored in the directory:

foxmail-installed-path/mail/FOXMAIL account number.

Four files are found under the Foxmail account number subdirectory: `in`, `out`, `send` and `trash`. These four files correspond to the email inbox, email outbox, email sent items and email deleted items, respectively.

According to the structured storage paradigm, all the emails in a mail folder are stored in one file. Our analysis revealed that the following byte sequence is used as the header for email in the mail file:

```
10 10 10 10    10 10 10 11 11 11 11 11 11 53 0D 0A
```

4.2 Forensic Analysis

The main task in Foxmail forensic analysis is to search for the email header and extract the mail content that follows the mail header [2]. Next, the extracted mail is exported to the EML format, which can then be processed by any email forensics tool.

5. Case Study

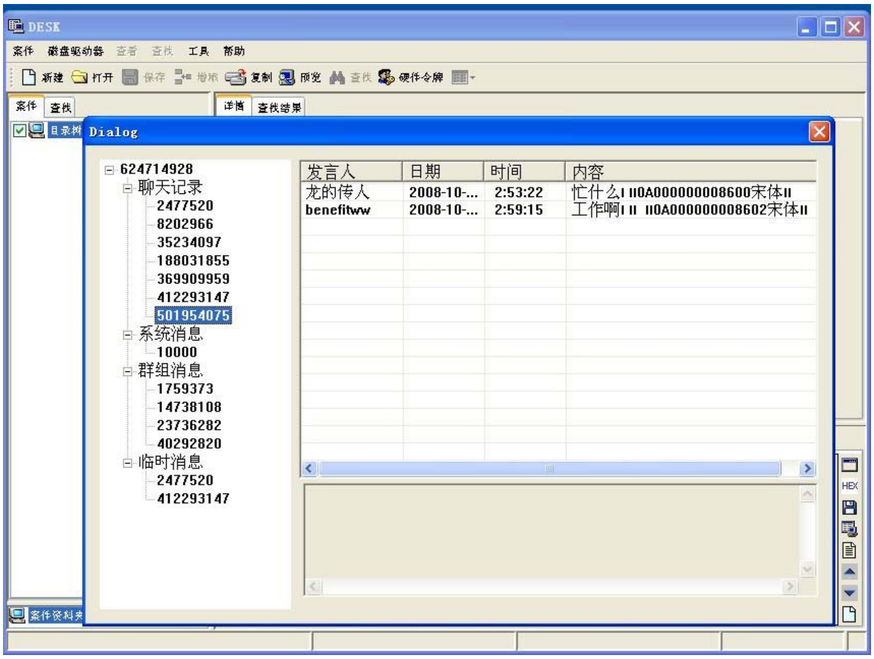
This section describes the use of Enhanced DESK in a case involving the theft of a computer.

A student named Mr. Zhang consigned a logistics company to deliver a computer to his home. The shipment was signed by his father upon arrival at his home. However, upon inspecting the contents of the shipment, Mr. Zhang discovered that his original computer was replaced by a cheaper machine, causing a direct financial loss of 6,500 Yuan.

Mr. Zhang contacted the company about the switch but got no results. He then lodged a complaint with the police, and the case was placed on file for investigation and potential prosecution. The Shangdong Computer Science Center was assigned to examine the computer and obtain digital evidence.

Enhanced DESK was used to clone the computer hard disks and create a backup. The examination of the forensic image revealed more than ten QQ accounts. Enhanced DESK was then used to extract QQ-related information. The recovered chat logs were saved to a `.txt` file for analysis. Photographs were taken of the entire process as subsidiary evidence. Finally, text and photographs from QQ Zone were collected based on information recovered about the owners of the QQ accounts.

Upon reviewing the digital evidence, the police officer assigned to the case confirmed that the suspect was an employee of another logistics


DESK
案件 磁盘驱动器 查看 查找 工具 帮助
新建 打开 保存 增添 复制 预览 查找 硬件令牌
案件 查找
详情 查找结果
Dialog
HEX
案件资料夹

```
624714928
  聊天记录
    2477520
    8202966
    35234097
    188031855
    369909959
    412293147
    501954075
  系统消息
    10000
  群组消息
    1759373
    14738108
    23736282
    40292820
  临时消息
    2477520
    412293147
```

发言人	日期	时间	内容
龙的传人	2008-10-...	2:53:22	忙什么I II0A000000008600宋体II
benefitww	2008-10-...	2:59:15	工作啊I II II0A000000008602宋体II

Figure 6. QQ forensic analysis using Enhanced DESK.

company. When confronted with the evidence, the suspect confessed to taking Mr. Zhang's computer and replacing it with a cheap substitute.

Figure 6 shows a sample Enhanced DESK screen dump during a forensic investigation involving the QQ application.

6. Conclusions

The enhanced version of DESK supports forensic investigations of major Chinese Internet communication applications such as QQ, MSN and Foxmail. Our future research will attempt to develop a lightweight version of DESK targeted for forensic investigations in the field.

References

[1] K. Chow, C. Chong, P. Lai, L. Hui, K. Pun, W. Tsang and H. Chan, Digital Evidence Search Kit, *Proceedings of the First International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 187–194, 2005.

[2] J. Feng, The implementation of Foxmail email converter by VC platform, *Computer Programming Skills and Maintenance*, vol. 10, pp. 45–46, 2003.

[3] Microsoft Corporation, Structured Storage, Redmond, Washington (msdn.microsoft.com/en-us/library/aa380369(VS.85).aspx).

[4] QQ International, QQ, Shenzhen, China (im.qq.com).

[5] R. Rivest, The MD5 Message-Digest Algorithm, IETF RFC 1321, 1992.

[6] Y. Shi and Y. Zhang, IM system model based on MSNP protocol, *Computer Engineering and Applications*, vol. 41(36), pp. 142-144, 2005

[7] R. Spillman, *Classical and Contemporary Cryptology*, Prentice-Hall, Upper Saddle River, New Jersey, 2004.

[8] Tencent Holdings, Foxmail, Shenzhen, China (fox.foxmail.com.cn).

Chapter 21

DATA RECOVERY FUNCTION TESTING FOR DIGITAL FORENSIC TOOLS

Yinghua Guo and Jill Slay

Abstract Many digital forensic tools used by investigators were not originally designed for forensic applications. Even in the case of tools created with the forensic process in mind, there is the issue of assuring their reliability and dependability. Given the nature of investigations and the fact that the data collected and analyzed by the tools must be presented as evidence, it is important that digital forensic tools be validated and verified before they are deployed. This paper engages a systematic description of the digital forensic discipline that is obtained by mapping its fundamental functions. The function mapping is used to construct a detailed function-oriented validation and verification framework for digital forensic tools. This paper focuses on the data recovery function. The data recovery requirements are specified and a reference set is presented to test forensic tools that implement the data recovery function.

Keywords: Digital forensic tools, validation, verification, data recovery

1. Introduction

Digital forensics is the process of identifying, preserving, analyzing and presenting digital evidence in a manner that is acceptable in courtroom proceedings [5]. As identified in [1, 2], one of challenges in the discipline is to ensure that the digital evidence acquired and analyzed by investigative tools is forensically sound.

In our previous work [2], we proposed a function-oriented framework for digital forensic tool validation and verification. The framework identified fundamental functions involved in digital forensic investigations such as search, data recovery and forensic copying. A process called "function mapping" was used to further identify the details of each function (e.g., sub-categories and components). The results enable the specification of the requirements of each function and help develop a reference set against which digital forensic tools may be tested.

Our previous work addressed the first task in creating a validation and verification framework, i.e., the "search" function. This paper attempts to address the second task – to complete the function mapping, requirements specification and reference set development of the "data recovery" function. The following sections review our function-oriented validation and verification framework, present the details of the data recovery function mapping, and describe a pilot reference set for testing the data recovery function.

2. Validation and Verification Framework

Our validation and verification framework [2] is function-oriented and incorporates detailed specifications that are absent in other work. The methodology begins with a systematic description of the digital forensic field using a formal model and function mapping. Digital forensic components and processes are defined in this model and fundamental functions in the investigative process such as searching, data preservation and file identification are specified (i.e., mapped). Having developed the model and function mapping, the validation and verification of a digital forensic tool is accomplished by specifying its requirements for each mapped function. Next, a reference set is developed comprising a test case (or scenario) corresponding to each function requirement. The reference set enables the forensic tool and/or its functions to be validated and verified independently.

This paper engages the CFSAP model [6] to describe the basic procedures involved in a digital forensic investigation: identification, preservation, analysis and presentation. In the context of validation and verification, identification and presentation are skill-based concepts. On the other hand, preservation and analysis are predominantly process-, function- and tool-driven concepts and are, therefore, subject to tool validation and verification.

Beckett and Slay [1] have dissected the processes of preservation and analysis into fundamental functions. Figure 1 presents a function categorization of validation and verification.

In this work, we attempt to complete the mapping of the functional categories of the digital forensics discipline at a level of abstraction that would serve the purposes of a specification for a software developer, technical trainer or educator; or for tool validation or verification. In particular, we detail the specification of function categories (e.g., searching, data preservation and file rendering) and their sub-categories. Our

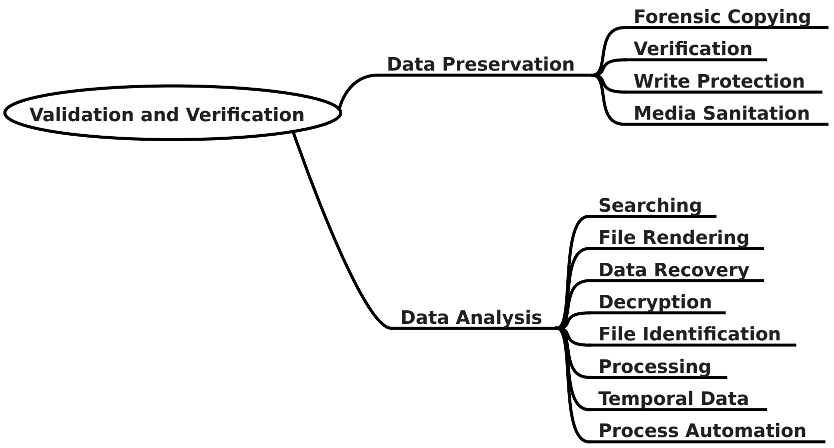


Figure 1. Validation and verification top-level mapping.

focus is on the data recovery function: mapping the function, specifying its requirements and developing the reference set to validate and verify tools that implement the data recovery function.

If the domain of digital forensic functions and the domain of expected results (i.e., requirements of each function) are known, in other words, the range and specification of the results are known, then the process of validating a tool can be as simple as providing a set of references with known results. When a tool is tested, a set of metrics can also be derived to determine the fundamental scientific measurements of accuracy and precision. In summary, if the discipline is mapped in terms of functions (and their specifications) and, for each function, the expected results are identified and mapped as a reference set, then any tool, regardless of its original design intention, can be validated against known elements. As claimed in [2], our function-oriented validation and verification regime has several distinctive features such as detachability, extensibility, tool version neutrality and transparency.

3. Data Recovery Function Mapping

Data recovery is generally regarded as the process of salvaging data partially or completely from damaged, failed, corrupted or inaccessible storage media. Recovery may be required due to physical damage to the storage device or logical damage to the file system that prevents it from being mounted by the host operating system.

A variety of failures can cause physical damage to storage media. CD-ROMs can have their metallic substrate or dye layer scratched off;

hard disks can suffer any of several mechanical failures; tapes can simply break. The logical damage to the data may take the form of corrupt or missing boot-related records (e.g., main boot record, disk partition table and directories) or the loss of file signatures (e.g., header and footer). Since our focus is on validating and verifying digital forensic tools in terms of the data recovery function, the consideration of physical damage recovery techniques is outside the scope of this paper and is considered to be complementary to logical damage recovery techniques. Consequently, in the rest of this paper, data recovery refers to logical damage recovery unless otherwise stated.

Data recovery in the context of digital forensics has its own peculiarities and differs from traditional data recovery in the computer science discipline. First, data recovery in the digital forensic context is a process by which digital evidence is recovered for use in court. Therefore, it should be conducted by certified investigators, conform to standard operating procedures, utilize tools that are validated and verified by the appropriate authorities, and be supervised and documented. Traditional data recovery does not have these requirements because its goal is to recover as much data as possible without concern for its forensic soundness. Second, the techniques used in traditional data recovery and in the digital forensic context differ because of the forensic soundness issue. For example, in traditional data recovery, a corrupted main boot record may be repaired by laying a FAT2 over a FAT1 if the FAT2 is intact. However, this is not an appropriate forensic data recovery technique because the original evidence (FAT1) is modified. Instead, it would be necessary to repair the corrupted main boot record in a duplicate (i.e., image). Finally, forensic data recovery embraces a broader view of recovering data than traditional data recovery and, consequently, must consider issues (e.g., hidden data and trace data) that are beyond the purview of traditional data recovery.

The data recovery function is mapped by detailing its components, processes and relevant factors. Since the goal of data recovery is to retrieve data due to storage media abnormalities and/or intentional human manipulation, the function mapping is performed from three angles: (i) storage media; (ii) recovery object; and (iii) recovery reason. Figure 2 presents the top-level ontology of the data recovery function.

3.1 Storage Media

Data is typically stored as files on storage media. The files are managed (i.e., created, modified and deleted) by file systems. In order to perform data recovery effectively and efficiently, forensic investigators

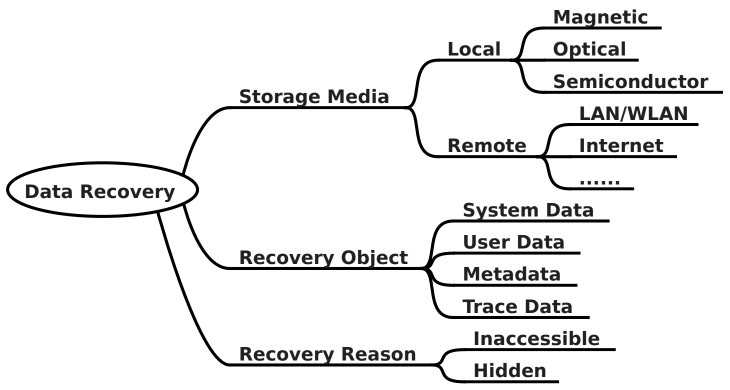


Figure 2. Top-level data recovery function mapping.

need to understand the physical media as well as the logical structures of files on the media.

Data recovery may be conducted locally (i.e., the storage media are seized and are under the custody of the investigator) or remotely (i.e., the investigator uses a network to access the storage media).

From the physical (material) point of view, storage media can be categorized as: magnetic, optical or semiconductor. The magnetic storage media category includes hard drives, RAID arrays, floppy disks, zip disks and tape drives (Figure 3). Typical hard drive types include ATA, SATA, SCSI, IDS and USB. The file systems used include FAT (12, 16, 32), NTFS, HFS (HFS+) and EXT (2, 3, 4).

Typical optical storage media are CDs and DVDs (Figure 4). CD storage media are in the form of CD-ROM, CD-R and CD-RW. Common file systems for CD media are ISO-9660, UDF, Joliet, HFS and HSG. DVD media include DVD-ROM, DVD-R(+R) and DVD-RW(+RW). The principal file systems for DVD media are UDF and HFS.

The principal semiconductor-based storage media are RAM and ROM (Figure 5). Flash memory, a type of EPROM (erasable programmable read-only memory), is widely used in computers and electronic devices and includes compact flash (CF) cards, smart media (SM) cards, secure digital (SD) cards, memory sticks and USB flash drives. File systems commonly used in flash memory include FFS, JFFS, LogFS and YAFS.

3.2 Reasons for Data Recovery

A data recovery method is used when data is unavailable. In the context of digital forensics, data is unavailable and must be salvaged for various reasons, including damage, corruption or hiding. From the

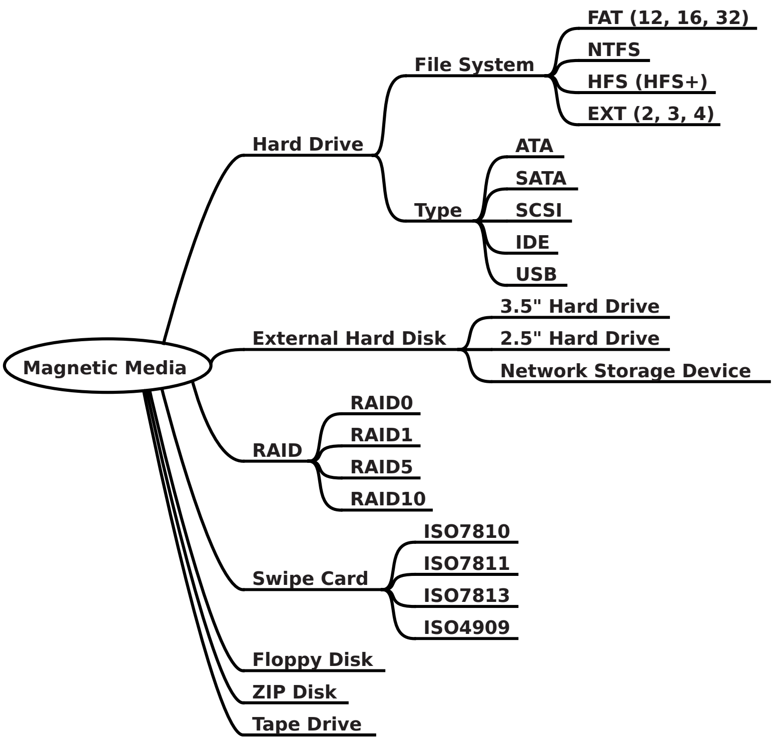


Figure 3. Magnetic storage media category.

point of view of the user (e.g., investigator), we assume that the data is unavailable because it is inaccessible or hidden. By inaccessible data, we mean that the user is aware of the existence of the data, but is unable to access it in a normal manner. On the other hand, hidden data is invisible to the user and the user does not know of its existence.

Inaccessible Data "Orphaned" files are inaccessible to users under normal operations. An orphaned file is one that no longer has a parent (the parent is the folder in which it was originally located) [4]. The term orphaned is a broad concept that includes deleted files. In most cases, orphaned files are deleted files, but a file can be orphaned when the association with its parent is lost through other means (e.g., by removing a symbolic link in a Unix environment).

Ambient space (unallocated space) or space that is orphaned from the operating system or file system has many forms. Data in such space cannot be accessed by users under normal operations. For example, file slack space is ambient or unallocated space that exists at the end of a file

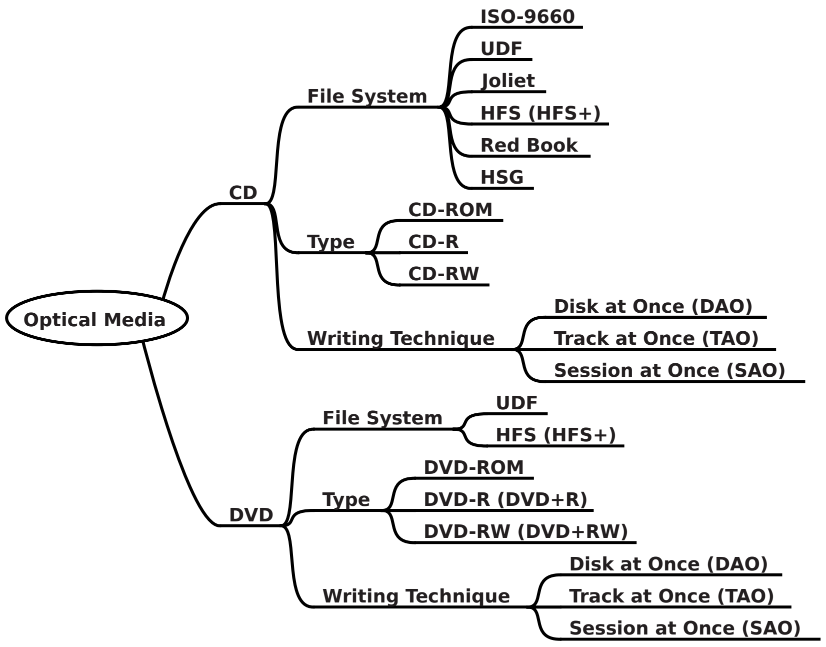


Figure 4. Optical storage media category.

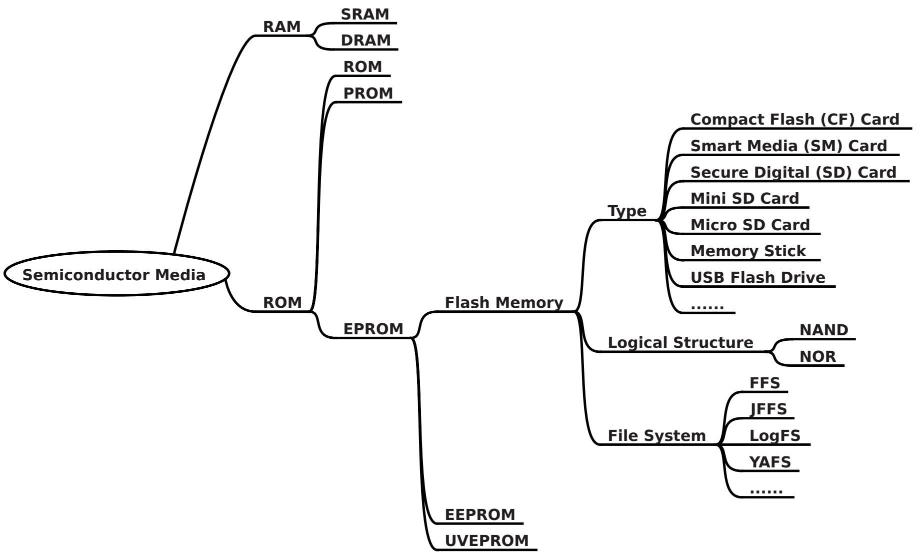


Figure 5. Semiconductor storage media category.

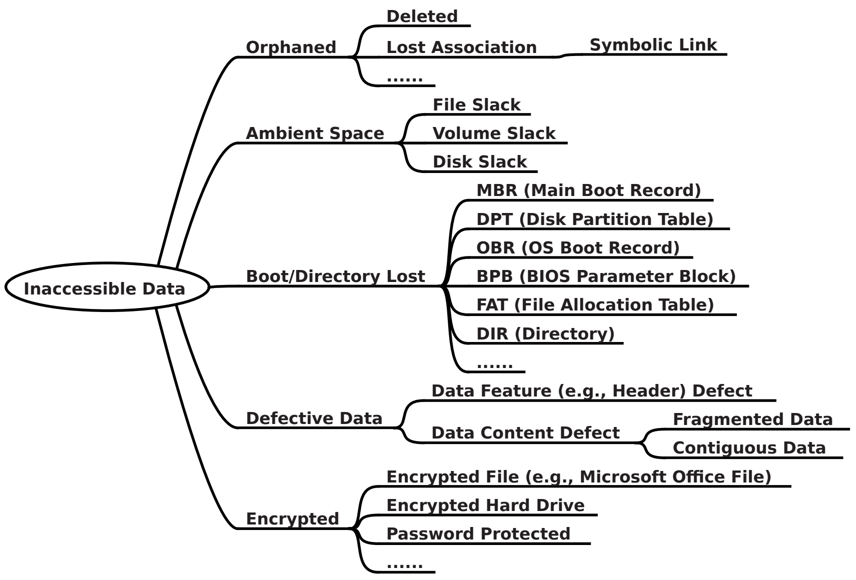


Figure 6. Inaccessible data category.

in certain operating systems and can contain a variety of data, including data dumped from RAM (RAM slack) or the remnants of previously-allocated files that may have been orphaned and partially overwritten.

Data may also be inaccessible because its metadata is corrupted or missing. The associated metadata includes MBR, DPT, OBR (operating system boot record), BPB (BIOS parameter block), FAT and DIR (directory). In such scenarios, the file may not be located, but its data is intact and, therefore, can be recovered by "file carving" [8].

Alternatively, a file can be located using metadata, but the data itself cannot be accessed because it is defective. This can occur for two reasons. One possibility is that the data feature (e.g., header or footer) is damaged. A file header is a "signature" placed at the beginning of a file to enable the operating system or application to know what to do with the following contents. The file cannot be recognized when this feature is damaged. The second possibility is that the data content is corrupted. In this case, it is necessary to analyze the structural characteristics and code of the damaged file to recover the data or portions of the data.

Finally, data may be inaccessible due to encryption and steganography. Although an encrypted file is visible to users, its contents are inaccessible without the key. Figure 6 summarizes the inaccessible data category.

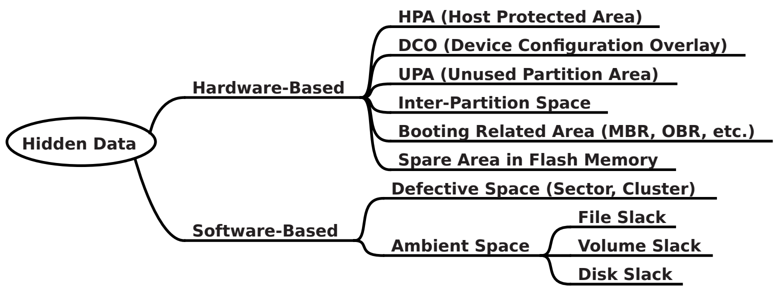

Hidden Data
Hardware-Based
HPA (Host Protected Area)
DCO (Device Configuration Overlay)
UPA (Unused Partition Area)
Inter-Partition Space
Booting Related Area (MBR, OBR, etc.)
Spare Area in Flash Memory
Software-Based
Defective Space (Sector, Cluster)
Ambient Space
File Slack
Volume Slack
Disk Slack

Figure 7. Hidden data category.

Hidden Data In the digital forensic context, it may be necessary to recover data that has intentionally been hidden. Data hiding methods may be categorized as hardware-based or software-based (Figure 7). Hardware-based methods hide data in specific areas of storage media. For example, data on a hard disk may be stored in the HPA (host protected area), DCO (device configuration overlay), UPA (unused partition area) and inter-partition space.

Software-based methods hide data using file system and/or operating system utilities [3]. For example, modern hard disk controllers handle bad sectors without the involvement of the operating system by slipping (modifying the LBN (logical block number) to physical mapping to skip the defective sector) or remapping (reallocating the LBN from a defective area to a spare sector). For older hard disks that do not have this capability, the operating system and file system have to retain the ability to detect and mark defective sectors and clusters as damaged. This feature can be used to exclude undamaged clusters from normal file system activities and use them to hide data.

Software-based data hiding methods may also use ambient space. Slack space, which includes file slack space, volume slack space and partition slack space, are areas on the disk that cannot be used by the file system because of the discrete nature of space allocation. Data can be hidden in any of these locations.

3.3 Recovered Objects

File system data to be recovered belongs to one of four categories: system data, user data, metadata and trace data.

System Data System data includes general hardware and software information. Data recovery techniques include hardware rendering and software (operating system and file system) rendering (Figure 8).

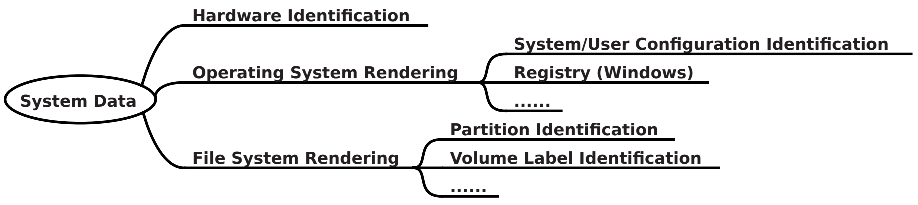


Figure 8. System data.

Hardware rendering refers to the ability to accurately identify particular types of devices and media. This is accomplished through physical interaction with the device or media or by using metadata located on the device or media.

The goal of operating system or file system rendering is to reveal the underlying structure. Operating systems and file systems have a general structure, but each instance is unique. In addition, many of these systems are proprietary in nature and, as a result, are poorly documented (e.g., the detailed structure of NTFS has not been publicly released). Operating system and file system rendering may specify where certain structures are found and the data unit size that enables file folders, data and metadata to be accurately retrieved. For example, the volume label and the associated data are indicators of the method used to create the allocated components of a device. Different file systems record this information differently, so a digital forensic tool must be able to render the volume label(s) from a device or partition.

User Data User data is the principal object of data recovery. User data are categorized as document, graphic, sound or Internet files. Figure 9 presents the classification and provides typical instances of each class. This classification is by no means exhaustive and will have to be updated constantly to accommodate new applications and file formats. Note that user data files may be in special forms (e.g., compressed and encrypted), which should be taken into account by forensic examiners.

Metadata Metadata is data that describes data or files. It includes data about where the file content is stored, file size, dates and times of the last read and write, and access control information. Figure 10 presents examples of metadata in various storage and file systems. Metadata must be analyzed to determine details about a specific file or to search for a file that meets certain requirements.

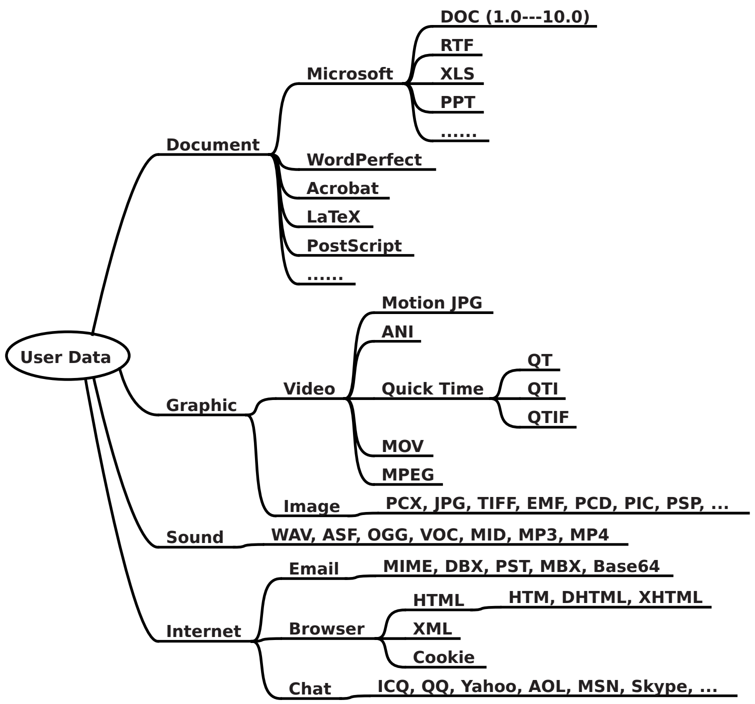


Figure 9. User data.

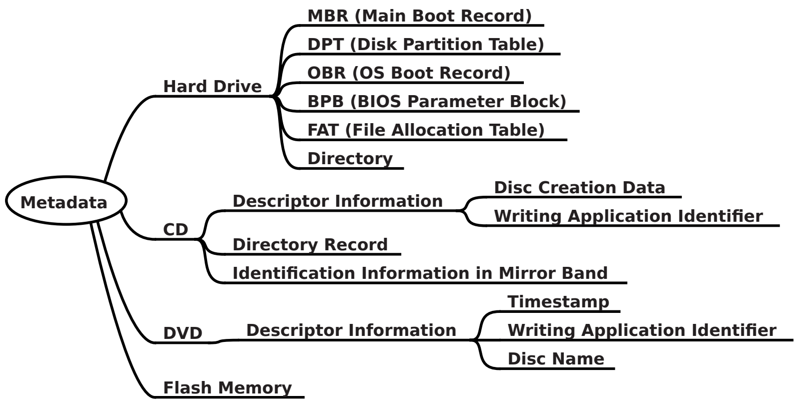


Figure 10. Metadata.

Trace Data As mentioned above, data recovery in the digital forensic context is a much broader concept than traditional data recovery. Trace data is the data that remains on the storage media after operations

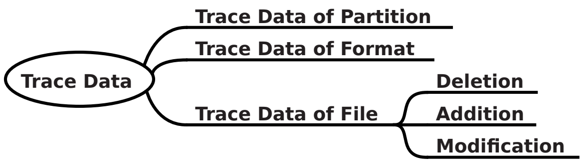


Figure 11. Trace data.

such as hard drive partitioning, formatting and file deletion (Figure 11). Trace data may not be substantial, but may constitute important digital evidence. For example, file operations (e.g., creation, modification and deletion) leave traces in the form of temporary files. Most temporary files are deleted by the operating system after the file operations are completed. However, if a temporary file is deleted, it contents can be recovered if the clusters allocated to the file are not reallocated. Also, even if the allocated clusters are reallocated, file metadata (e.g., name and timestamp) may exist and may prove to be useful in a digital forensic investigation.

4. Requirements Specification

Requirements specification is the second step of the validation and verification framework. The data recovery function requirements are specified in the same way as the search function requirements in [2].

The requirements are specified in an extensible and customized manner. As seen in function mapping, several issues have to be considered when specifying the requirements. For example, the storage media could be a hard disk, CD/DVD, flash memory, etc. The file system that manages data files on the storage media could be FAT12, FAT16, FAT32, EXT2, EXT3, NTFS, HFS(+), FFS, etc. The data could be inaccessible for any number of reasons; it could be orphaned, corrupted, encrypted, etc. Each of these sub-categories again has many variations.

The method of specifying requirements is highly abstract and generalized. We use italicized "variables" to reflect these variations. Thus, when a requirement has to be changed, it is only necessary to adjust (add, delete or modify) the variables. Moreover, the requirements can be unwrapped when it is necessary to develop a specific test scenario in a reference set. For example, the requirement: "The tool shall be able to accurately recover *inaccessible recovery objects*" may be unwrapped and instantiated as "The tool shall be able to accurately recover *deleted JPG files*" or "The tool shall be able to accurately recover *hidden data in file slack*."

A digital forensic tool has the following eight requirements with respect to the data recovery function:

- The tool shall operate in at least one *operational environment.*
- The tool shall operate under at least one *operating system.*
- The tool shall operate on at least one type of *storage media.*
- The tool shall be able to accurately render *system data.*
- The tool shall be able to accurately recover *inaccessible (recovery) objects.*
- The tool shall be able to accurately recover *hidden (recovery) objects.*
- If there are unresolved errors when reconstructing data, then the tool shall report the *error types* and *error locations.*
- The tool shall report the *attributes* of the recovered data.

5. Reference Set Development and Testing

A reference set consists of test scenarios (cases) against which a digital forensic tool or its individual function is validated. The development of test scenarios is based on the specification of function requirements. Using the requirements specification, it is possible to establish a reference set for testing the data recovery function of various digital forensic tools. Since the function requirements are specified in an extensible manner, the corresponding reference set is also extensible. This would enable practitioners, tool developers and researchers to identify critical needs and to target deterministic reference sets.

We have identified eight requirements for the data recovery function. Since each requirement has several variables, multiple test scenarios have to be designed for each requirement. Each scenario represents a single instantiation of each variable. The following are some pilot samples of the reference set for the data recovery function:

- A deleted JPG file in a FAT32 file system on an IDE hard disk.
- A deleted JPG file in an NTFS file system on a SCSI hard disk.
- A deleted WAV file in a UDF file system on a CD.
- A deleted Microsoft Word file in an FFS file system on flash memory.

- A deleted compressed HTML file in an NTFS file system on an IDE hard disk.
- An encrypted MP3 file in a FAT32 file system on an ATA hard disk.

Thus far, we have completed the function mapping, requirements specification and reference set development. We now know what needs to be tested and what the expectations are. Validating a digital forensic tool that professes to have a search function is now as simple as testing the tool against the reference set and applying metrics (accuracy and precision) to determine the quality of the results.

6. Conclusions

Mapping the fundamental functions of the digital forensic discipline is a powerful approach for creating a function-oriented validation and verification paradigm for digital forensic tools. The utility of the approach is demonstrated in the context of the data recovery function via the specification of data recovery requirements and a reference set for testing tools that implement the data recovery function. Validating a digital forensic tool is reduced to testing the tool against the reference set. Compared with traditional testing methods, this testing paradigm is extensible, and neutral and transparent to specific tools and tool versions.

More work remains to be done to complete the validation paradigm. Although the methodology holds promise, it needs to be tested extensively to evaluate its utility and identify potential weaknesses and shortcomings. Tests would have to be implemented against popular tools such as EnCase and FTK. A quantitative model is also required to evaluate the results of validation and verification. Metrics are needed to measure the accuracy and precision of testing results, and it is necessary to specify rules for judging the validity of digital forensic tools. Is a tool validated only when it passes all the test cases? Or is a tool validated when it passes the test cases for certain scenarios?

It is important to recognize that numerous variables are involved in function requirements specification and that the corresponding reference set can be very large. Indeed, the number of possible combinations for validating a single function in a digital forensic tool may well be in the thousands (even discounting the different versions of the tool). Interestingly, this problem is also faced by the Computer Forensics Tool Testing (CFTT) Program [7] created by the National Institute of Standards and Technology (NIST) to validate and verify digital forensic tools. This problem will be examined in our future work.

References

[1] J. Beckett and J. Slay, Digital forensics: Validation and verification in a dynamic work environment, *Proceedings of the Fortieth Annual Hawaii International Conference on System Sciences*, p. 266, 2007.

[2] Y. Guo, J. Slay and J. Beckett, Validation and verification of computer forensic software tools – Searching function, *Digital Investigation*, vol. 6(S1), pp. S12–S22, 2009.

[3] E. Huebner, D. Bem and C. Wee, Data hiding in the NTFS file system, *Digital Investigation*, vol. 3(4), pp. 211–226, 2006.

[4] D. Hurlbut, Orphans in the NTFS world, AccessData, Lindon, Utah (www.accessdata.com/media/en_US/print/papers/wp.NT_Orphan_Files.en_us.pdf), 2005.

[5] R. McKemmish, What is forensic computing? *Trends and Issues in Crime and Criminal Justice*, no. 118 (www.aic.gov.au/publications/tandi/ti118.pdf), 2002.

[6] G. Mohay, A. Anderson, B. Collie, O. de Vel and R. McKemmish, *Computer and Intrusion Forensics*, Artech House, Norwood, Massachusetts, 2003.

[7] National Institute of Standards and Technology, Computer Forensics Tool Testing Program, Gaithersburg, Maryland (www.cftt.nist.gov).

[8] A. Pal and N. Memon, The evolution of file carving, *IEEE Signal Processing*, vol. 26(2), pp. 59–71, 2009.