1. INTRODUCTION

Computer forensics is a multidisciplinary field concerned with the examination of computer systems which have been involved in criminal activity, either as an object or a tool of a crime. The aim of the investigator is to find information relevant to the case in question, as well as the chain of events leading to the creation of this information. In other words the questions to be answered are “What incriminating information is present in the system?” and “How did the incriminating information get there?”

How hard or easy it is to answer these questions depends in all cases on how the information of interest is stored by the operating system (i.e. the internal structure), and the analysis tools the operating system provides (i.e. the functionality). Mainstream modern operating systems were designed with many objectives in mind, for example performance, flexibility, expandability, user friendliness, and more recently security. They were not specifically designed to be forensically friendly, and forensic investigators struggle with the obstacles this creates in their daily practice.

Computer crime is on the increase, and with computer technology pervading all aspects of human activity, this problem will be more and more acute. Operating systems, and the file systems they support, could be intrinsically designed and implemented in a way which makes forensic investigation less time consuming and more reliable, and assists the use of the uncovered evidence in prosecution of perpetrators. Currently there are many external software tools which investigators use, and all of them rely on the interface the operating system provides. In a sense these tools’ access to the information is “second hand”. For example, even when dealing with a fundamental issue such as post-mortem examination of hard disk images, the available information depends on how the time stamps on files were handled in a live system, what was the process of deleting or overwriting files, which file system events were logged etc.

The rules of evidence, which determine the admissibility of findings in the court of law, demand that the accuracy of the methods used to collect the evidence is known, and the evidence is not tampered with in the process of its analysis. Unfortunately even the traces of the simplest facts such as time of file access or an event logged by the system, are not as trustworthy as required by the rules of evidence. Some believe that hardware offers the only guaranteed method of collecting evidence from computer systems. This belief can only change if the underlying software (operating system and file system) creates and maintains information in a more forensic-friendly structured and reliable way.

The above state of affairs is the main motivation behind this special issue of the Operating Systems Review, which focuses on the relationship between computer forensics and operating systems design and implementation. Our hope is to boost interest in forensic aspects of systems software design and implementation, as well as to demonstrate that computer forensics is a worthwhile direction for further research.

2. THE PAPERS

Papers selected for this issue deal with a wide range of problems in computer forensics, all of which are related in various ways to operating systems. They present a variety of views on how operating systems hinder or support forensic investigation, as well as ideas on how they could potentially support it better. The topics cover operating systems instrumentation (including the means for counteracting anti-forensic tools), operating system support for data acquisition and analysis, computer forensics of virtual systems, cryptography in forensic analysis, network forensics, and a discussion of computer forensics as a discipline of science.

The content of this special edition was determined following a rigorous process involving peer review, revision, and further review. Authors initially submitted papers for consideration in response to a general call. Each paper was peer reviewed by at least three reviewers, after which selected authors were invited to revise and re-submit. The final set of papers represents less than half of the proposals initially received by the editors.

To assist the reader, the editors have grouped the accepted papers according to the area of computer forensics they address. These groupings, together with a brief summary of each paper’s content, follows.

2.1 Operating System Instrumentation for Forensics

Although there is much to be done to make operating systems more forensically friendly, the existing forensic techniques do not take full advantage of existing system capabilities.

One such example is the journaling feature of the Ext3 file system. Xia, Fairbanks, and Owen propose a forensics enabling architecture which provides a means of using the metadata inherent in the Ext3 file system to reconstruct probable sequences of events that occurred during the journaling process. The reconstruction procedure is achieved by generating program behavior signatures produced by system call monitoring tools built into the operating system. These signatures could allow forensic investigators to perform probabilistic analysis based on
information theory models to extract a more significant set of data.

As computer forensics moves towards acquiring not only permanent but also volatile storage, new problems arise with the soundness of the techniques used. The addition of a memory acquisition mechanism to the operating system, proposed by Libster and Kornblum, removes the need to load an external program to accomplish this task. The method minimizes the impact of memory acquisition on the system’s state, at the same time making it more difficult for malicious programs to avoid detection or interfere with the memory dump. The risks of allowing a full memory capture, as well as some considerations on how this method would interact with rootkits, are also discussed.

The analysis of a compromised system is a time consuming and error-prone task because commodity operating systems provide limited auditing facilities. Goel, Farhadi, Po and Feng have been developing an operating-system level auditing system that captures a high-resolution image of all system activities so that detailed analysis can be performed after an attack is detected. The challenge with this approach is that the large amount of generated audit data can overwhelm analysis tools. Their paper describes a technique that helps generate a time-line of the state of the system. This technique, based on preprocessing the audit log, simplifies the implementation of the analysis queries and enables running the analysis tools interactively on large data sets.

YasinSac and McDonald examine and set forth principles of operating system design that have a potential to significantly increase the success of future forensic collection efforts. They formulate a number of operating system design attributes that work together to enhance forensic activities. Specifically, they propose the use of circuit encryption techniques to create an additional layer of protection above hardware-enforced approaches. They conclude by providing an overarching framework to incorporate these enhancements within the context of operating system design.

Monteiro and Erbacher present a method of logging events that ties together, authenticates, and validates all the entities involved in the crime scene: the user using the application, the system that is being used, and the application being used on the system by a user. This means that instead of merely transmitting the header and the message, which is the standard syslog protocol format, the syslog entries along with the user fingerprint, application fingerprint, and system fingerprint are transmitted to the logging server. System logs generated this way are validated forensically by assignment of digital fingerprints and the addition of a challenge response mechanism to the underlying syslogging mechanism.

The inevitable response to the development of standard forensic procedures and tools is a growth in anti-forensic techniques. Maggi, Zanero and Iocco propose the use of machine learning algorithms and anomaly detectors to circumvent anti-forensic attacks. They present a prototype of anomaly detector which analyzes the sequence and the arguments of system calls to detect intrusions. They demonstrate the use of this prototype to detect in-memory injections of executable code, and in-memory execution of binaries. The detector creates a usable audit trail, without the need to resort to complex memory dump and analysis operations.

2.2 Operating System Support for Data Acquisition and Analysis

Data acquisition and analysis lies at the heart of computer forensics. It is the operating system which has the most direct access to all data in a computer system, as it has either created or supported the creation of this data.

A system calls monitoring approach for detection of kernel rootkits has been proposed by Wampler and Graham. Their approach focuses on detection using reduced a priori knowledge in the form of general understanding of the statistical properties of broad classes of operating system/architecture pairs. They show that using modified normality proved effective in detecting kernel rootkits infecting the kernel via the system call target modification attack. This approach capitalizes on the discovery that system calls are loaded into memory sequentially, with the higher level calls, which are more likely to be infected by kernel rootkits loaded first, and the lower level calls loaded later. In the single case evaluated, the enyelkm rootkit, neither false positives nor false positives were indicated.

The rapid changes in memory development and technology have a major impact on forensic methodology. Operating systems, and their revisions, handle internal data in different ways and so new tools are being developed and existing tools and techniques revised rapidly to keep in step with these changes. It is therefore important to have the ability to test forensic tools in a manner which is flexible and resilient to change. Specifically, it should be possible to develop a methodology to aid the investigator in assessing the forensic impact of tools on a target’s operating system. Sutherland, Evans, Tryfonas and Blyth examine current practice in the collection of live memory artifacts and assess the impact of particular tools and techniques in the acquisition of live memory. Their paper analyses Windows XP Service Pack 2 systems on the Intel architecture and proposes a series of metrics to allow an examiner to quantify the forensic footprint of any given tool.

2.3 Forensics and Virtual Systems

Involving the live analysis of target systems to uncover volatile data presents significant risks and challenges to forensic investigators, as observation techniques are generally intrusive and can affect the system being observed. Hay and Nance discuss the issues of live digital forensic analysis through virtual introspection, and present a suite of virtual introspection tools developed for Xen (VIX tools). The VIX tools suite can be used for unobtrusive digital forensic examination of volatile system data in virtual machines It also addresses a key need of researchers in the area of virtualization in digital forensics.

Franklin, Luk, McCune, Seshadri, Perrig and Doorn studied the remote detection of virtual machine monitors (VMMs) across the Internet, and devised fuzzy benchmarking as an approach that can successfully detect the presence or absence of a VMM on a remote system. Fuzzy benchmarking works by making timing measurements of the execution time of particular code sequences executing on the remote system. The fuzziness comes from heuristics which the authors employed to learn characteristics of the remote system’s hardware and VMM configuration. Their techniques were successful despite uncertainty about the remote machine’s hardware configuration.
2.4 Cryptography in Forensic Analysis
The integration of strong encryption into operating systems is creating challenges for forensic examiners, potentially preventing recovery of any digital evidence from a computer. Casey and Stellatos present the evolution of full disk encryption (FDE) and its impact on digital forensics. Furthermore, by demonstrating how full disk encryption has been dealt with in past investigations, they provide forensic examiners with practical techniques for recovering evidence that would otherwise be inaccessible.

2.5 Network Forensics
Passive monitoring of the data entering and leaving an enterprise network can support a number of forensic objectives. McHugh, McLeod and Nagaonkar have developed analysis techniques for NetFlow data that use behavioral identification and can confirm individual host roles and behaviors expressed as connection patterns. By looking at the way a given machine interacts with others, it is often possible to determine the role of the machine based solely on such network data. Host behaviors as characterized by NetFlow data are not stationary. Evolutionary changes occur as the result of new applications, and computational and communications paradigms. Compromised machines often undergo changes in behavior that range from subtle to dramatic. Behavioral changes can be used to identify role shifts and to trace the malicious or unintentional propagation of that change to other machines. Observed behavioral characteristics from over a year of traffic captures containing ordinary behaviors as well as a variety of compromises of interest are presented as examples for the forensics practitioner or researcher.

2.6 Computer Forensics as a Discipline of Science
In 1993 a legal precedent was set by the U.S. Supreme Court regarding the admissibility of expert witnesses' testimony. This precedent came to be known as the Daubert test (see U.S. Supreme Court Ruling issued on 28 June, 1993). In the Daubert ruling the U.S. Supreme Court suggested four criteria for determining whether science was reliable and, therefore, admissible:
- Is the evidence based on a testable theory or technique?
- Has the theory or technique been peer reviewed?
- In the case of a particular technique, does it have a known error rate and standards controlling its operation?
- Is the underlying science generally accepted?
These four criteria are commonly used to assess the evidence obtained by any forensic methodology and tools, including computer forensics.

The development of computer forensics as a legitimate discipline of science requires that all involved parties communicate and work together using the same language and conceptual frameworks. This is the topic of the last paper by Peisert, Bishop and Marzullo, which is particularly suitable to conclude this collection. The paper presents several forensic systems and discusses situations in which they produce valid and accurate conclusions, as well as situations in which their accuracy is suspect. Further, forensic models are analyzed with the focus on areas in which they are useful and areas in which they could be augmented. The recommendation is that computer scientists, forensic practitioners, lawyers, and judges should build more complete models of forensics that take into account appropriate legal details and lead to scientifically valid forensic analysis. The authors conclude that it is the ability to communicate about the challenges of each side that will ultimately help bring scientific method to computer forensics in the way that it exists in other forensic disciplines.

3. ACKNOWLEDGMENTS
First and foremost we would like to thank the authors of all submitted papers, as well as those we were not able to include in this issue, for their time and effort.

We thank our colleagues on the Editorial Committee who evaluated the submissions at a time of year when others take a well deserved holiday. Listed in alphabetical order, the Editorial Committee is:
- Derek Bem (University of Western Sydney)
- Ljiljana Brankovic (University of Newcastle)
- Bill Caelli (International Information Security Consultants)
- Brian Carrier (Basis Technology)
- Al Dearle (University of St Andrews)
- Michael Hannaford (University of Newcastle)
- John Rosenberg (Deakin University)
- Dave Munro (University of Adelaide)
- Magdalena Szezynska (Warsaw University of Technology)

Special thanks go to John McHugh (Dalhousie University) for solving the formatting problem in the Latex template in the proverbial eleventh hour.

Finally, we thank the Operating Systems Review editor, Professor Jeanna Mathews of Clarkson University and ACM SIGOPS for providing this opportunity for the computer forensics researchers and practitioners to make our work more widely known to the operating systems community, and hopefully increase the general interest in forensics and its implications in their field.