

Towards High Assurance Networks of Virtual Machines

Fabrizio Baiardi¹, Daniele Sgandurra²

¹Polo G. Marconi La Spezia, Università di Pisa, Italy

²Dipartimento di Informatica, Università di Pisa, Italy

{baiardi, daniele}@di.unipi.it

1 Introduction

Any approach to intrusion detection should take into account that most sophisticated attacks strive to occupy the lowest system level. This is due to at least two reasons: (i) to achieve a full system control; (ii) to deceive the legitimate system owner by hiding the traces of the compromise and providing the owner with a false sense of security. As an example, *rootkits* have gradually evolved from user-level to kernel-level rootkits [10, 30]. This implies that, since both rootkits and tools that detect them run at the lowest level, the kernel one, only the first one able to predict and prevent the adversary moves can eventually succeed in getting full control of the system [27].

In the context previously described, a fully general, level-independent technique to discover signs of intrusions is *introspection*. This technique analyzes the system memory to rebuild data structures of interest and check their consistency. If no dedicated hardware support is available [26], *virtual machine introspection* [13] can be applied provided that the OS and applications to be monitored are executed inside a VM. A VM is an execution environment created by a virtualization technology that emulates, at software, the behavior of the underlying architecture [7, 34]. To create, manage and monitor the VMs, this technology introduces the *virtual machine monitor* (VMM) [12] a thin software layer in-between the OS layer and the hardware/firmware one. In this way, a standard physical machine supports several VMs, each running a distinct OS. By exploiting the VMM direct access to the memory of each VM, virtual machine introspection can analyze the state of the kernel hosted on a VM. In this way, introspection is applied at a lower level than the one an attacker can gain and, provided that the VMM cannot be subverted, we can build from the VMM up a chain of trust where each level applies a set of different consistency checks to discover attacks. In the following we will use the term introspection rather than the more correct, but longer, virtual machine introspection.

As discussed in the next sections, we believe that VMM and introspection are important building blocks to create high assurance systems. In this view, we have developed *Psyco-Virt*, a software architecture that integrates introspection with a set of host and network IDS tools to achieve high assurance on the integrity of the VMs. The overall architecture consists of a cluster of monitored VMs (Mon-VMs), i.e. the VMs to monitor, and introspection VMs (IVMs) to implement the monitoring. All the Mon-VMs are mapped onto a cluster of physical nodes, and one IVM is introduced for each physical node. All the Mon-VMs are connected by a virtual network, the data one, to exchange application traffic. A further virtual network, the *control network*, connects all the IVMs and each IVM and the Mon-VMs on the same node. This is a private hierarchical network that spans across distinct physical nodes to support the exchange of alerts and introspection information. A set of *IDS agents* on each Mon-VM discovers attempted intrusions/attacks and, in such a case, an agent alerts the IVM through the control network. In *Psyco-Virt*, the kernel of each Mon-VM guarantees the integrity of the controls implemented by the IDS agents, while an *introspector* running on the IVM exploits the VMM control interface to apply introspection and monitor the kernel of each Mon-VM to discover attacks against the kernel itself. In this way, the overall assurance is achieved in three steps:

1. the IDS agents apply a broad range of security checks, such as file system integrity, denial-of-service detection and prevention, anomaly detection;
2. the kernel of a Mon-VM controls the correct execution and integrity of the IDS agents;
3. the IVM applies introspection to assure the integrity of the kernel inside each Mon-VM.

Obviously, the second step may also use already existing security features, such as those offered by SELinux [21, 22].

The overall architecture provides assurance that critical VM components, such as the kernel and the agents, cannot be tampered with to make them return bogus data. In this way, the *trusted computing base* (TCB) includes all the components protected by the chain of trust starting with introspection. In turn, this strongly reduces the probability of a successful attack against the Mon-VMs. A further advantage of the architecture is that the configuration of each VM can be specialized to minimize the software it runs according to both its role and the applications of interest [33, 28, 2]. Lastly, since IDS agents may be simple wrappers to standard IDS tools, the architecture exploits at best current security tools to discover intrusions and attacks against the Mon-VMs.

The rest of the paper is organized as follows. Sect. 2 describes the overall architecture of Psycho-Virt and justifies the main assumptions underlying its definition. Sect. 3 discusses the current implementation and shows some examples of IDS agents and of integrity checks applied at different levels. Sect. 4 presents an evaluation of security and performance results and describes the current limitations of the prototype. Sect. 5 discusses related works. Finally, in Sect. 6 we draw a first set of conclusions and outline future developments.

2 Psycho-Virt Overview

After presenting the overall architecture of Psycho-Virt, we discuss the tasks of the IVMs and of the Mon-VMs.

2.1 Overall Architecture

Psycho-Virt assumes that the applications of interest are mapped onto a cluster of VMs, the Mon-VMs, which are then mapped onto a cluster of physical nodes. Each node runs a VMM and an introspector VM (IVM). In

turn, each Mon-VM runs an OS and a set of IDS agents (see Fig. 1). A *collector* on each Mon-VM supports the exchange of alerts and commands among the agents and the IVM. All the Mon-VMs are connected by a data network that supports application and OS traffic. A distinct network, the control one, connects the IVMs and each IVM to the Mon-VMs on the same node. This is a virtual private network, which cannot be accessed from the outside world. The control network connects the Mon-VMs in a hierarchical way to the IVMs to exchange: (i) commands among the IVM and the collector or the agents; (ii) alerts reporting intrusions or attacks from the agents to the IVM; (iii) control information among the IVMs to coordinate the detection.

One of the main tasks of an IVM is to apply introspection to protect the kernel of each Mon-VM. In this way, Psycho-Virt integrates both agents to detect intrusions on the Mon-VMs and introspection to preserve the integrity of the kernel of the Mon-VMs themselves. Each agent is a wrapper for a tool, such as chkrootkit [6] or Snort [31], which monitors critical parts of the system and alerts the IVM through the control network each time an intrusion is suspected. The IVM analyzes the alerts received from the agents and handles the detection of an intrusion or attack by executing a proper action, i.e. stop or freeze the execution of a compromised Mon-VM. The ability of freezing a VM to examine its state in more details is a fundamental advantage and a peculiarity of the virtualization technology.

Note that a Mon-VM can be introduced just to run an agent, as an example, a network IDS that analyzes the data network traffic. Obviously, the configuration of this Mon-VM should be optimized to reduce the software it runs and hence the opportunity of a successful attack against it.

To justify the multi-level approach we have adopted, consider an alternative solution that applies introspection by modifying an existing IDS tool so that it can apply introspection at the VMM level from the IVM. The complexity of this solution is very high because the IDS tool should monitor the Mon-VMs through a set of checks defined at the hardware level. A further, distinct, approach applies introspection to evaluate a set of consistency checks defined according to the OS-level semantics. This approach does not have to modify the IDS, provided that the introspection library enables the IDS to exploit a high level view of the VMs, defined in terms of files, processes and virtual memory. This is particularly challenging, as in the case of reproducing the structure and the operations of an existing file system. Both approaches are feasible, but the first one requires current IDS tools to be modified so that they work at the hardware level. On the other hand, the second approach requires a complex introspection library that should also support the various versions of the Oses of interest. Our approach is different from both the previous ones. In fact, Psycho-Virt dis-

covers intrusions on a Mon-VM through agents that exploit standard IDS tools, which do not have to be modified, as in the first approach. Furthermore, to simplify the adoption of introspection, our solution extends the kernel with functions that monitor the agents. As a consequence, Psycho-Virt only requires an introspection library much simpler than the one of the second approach because it only evaluates kernel related properties, and the introspector on the IVM can guarantee the integrity of the overall detection system by monitoring the kernel of the Mon-VMs only.

An important assumption of our approach is that the VMM can be trusted and that introspection is the basis of a chain of trust from the VMM to the kernel and then to the IDSes. The reason for assuming that both the VMM and introspection are trusted is twofold. Firstly, the VMM has full visibility of a Mon-VM, because it can access every VM components, such as the memory it allocates to them. Secondly, the VMM is more robust than commodity OSes because: (i) the interface it exports to the higher levels is very simple and so more difficult to subvert than, for example, the one of an OS kernel that implements hundreds of system-calls; (ii) the small size of its code reduces the likelihood of a compromise. In conclusion, since the VMM has full visibility of the VMs it supports and it is essentially isolated from these VMs, the complexity of compromising the VMM or of eluding the introspection monitoring capabilities is very high.

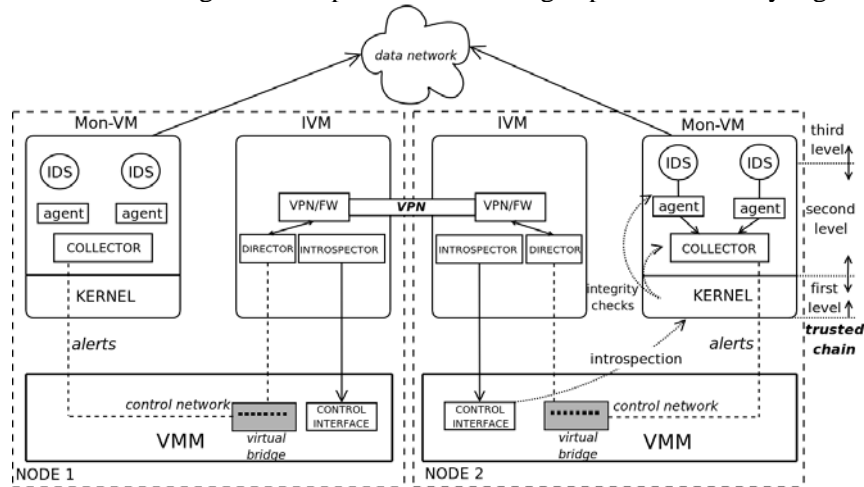


Fig. 1. Psycho-Virt Architecture

2.2 Introspection VM

The introspector on the IVM applies introspection to discover attacks against critical components of the kernel of a Mon-VM. In particular, it monitors the text section of the kernel and that of the loaded modules to discover whether they have been modified. These memory regions should be read-only, hence any attempt to update them implies that an attacker is trying to insert and execute arbitrary instructions. As an example, this happens when a kernel-level rootkit tries to modify the code of a system call.

When an agent on a Mon-VM detects an attack or when the introspector discovers that the kernel of a Mon-VM has been modified, the IVM handles the corresponding event by executing one of the following actions: (i) stop the execution of a VM and save its state in a file; (ii) kill a process (the offending one); (iii) close the connection of the VM to the data network; (iv) disconnect a user connected to a VM; (v) send an alert to the system console to inform the administrator. Since further information may be required to choose the proper action, the IVM runs a *director* to: (i) collect all the alerts from either the agents through the collector or the introspector; (ii) actively interact with the agents on the Mon-VMs to access specific information about the monitored systems.

The Psycho-Virt introspection library enables the introspector on the IVM to build a high level view of a Mon-VM state starting from the raw data in the Mon-VM memory. This library plays a fundamental role because the VMM control interface provides only a low level view of a Mon-VM, i.e. in terms of memory, registers and disk blocks, whereas the introspector should reason in terms of kernel data structures, such as the process descriptor or the system call table. Thus, the library should translate the low level data received from the VMM control interface into a high level view in terms of kernel data structures. This requires that the library knows the kernel hosted by each Mon-VM, the data structures it uses and the memory areas where they are allocated. In this way, Psycho-Virt can exploit a kernel level view of the status of the Mon-VMs and consider global information such as the list of the running processes, the list of the loaded modules or information associated with a process identifier (PID), such as the list of open files/sockets. This information may be used to implement a very general and effective consistency check: the IVM builds a list of kernel data structures and compares this information against the one returned by an OS command executed on the Mon-VM. Any difference may signal that an attacker is trying to hide her presence.

The control network among the IVMs simplifies the recognition of distributed attacks because the IVMs can broadcast information about attacks, detected either by agents or through introspection. This network is also

used to coordinate the shutdown of the Mon-VMs executing applications that are the target of a distributed attack.

2.3 Monitored VM

Each Mon-VM may run several IDS agents, each checking a distinct aspect of the OS or of the applications. The collector receives all the messages from these agents, and immediately forwards any message that reports an attack or an intrusion to the director on the IVM. Custom agents may be created to directly interact with Psycho-Virt to discover intrusions, or existing NIDS/HIDS tools can be used. For example, custom agents could analyze audit logs to discover unsuccessful login attempts. Conversely, tools such as Tripwire [25] may alert the collector any time they detect an attempted intrusion or attack. Each agent behaves like a common host IDS that monitors system calls, audit and applications log files and file system changes. The number of agents and the checks they implement are strongly related to the required security level.

3 Current Prototype

The first prototype of Psycho-Virt has been developed using the C language. The VMM we used is Xen [7] while all the Mon-VMs run the Linux Debian distribution. We also used XenAccess [36] as the basis of the Psycho-Virt introspection library and the Xen Control library to stop a VM, save its state to a file and resume a suspended VM. Lastly, we used OpenSSL [23] and the Linux Cryptolib to compute the hashes, while OpenVPN [24] was used to set up the control network among the IVMs in distinct nodes. The source code of the current prototype is available at <http://www.di.unipi.it/~daniele/projects/projects.php>.

3.1 Introspection Functions

The following paragraphs discuss the implementation of some sample introspection functions to exemplify some of the capabilities of Psycho-Virt. To implement these functions, we used the headers of the Linux kernel running inside the Mon-VMs, so that through introspection the introspector could reconstruct kernel data structures.

3.1.1 Detecting Kernel Modifications

This introspection function discovers attacks to the kernel of a Mon-VM. The IVM checks the memory pages storing:

- the kernel code, from the address *_text* to *_etext*;
- the system call dispatch table, stored in the *sys_call_table* array;
- the interrupt descriptor table, stored in the *idt_table* table.

Since these pages should never be modified, the introspector periodically computes their hashes to verify that they have not been updated. To detect illegal updates to pages that can be modified, a set of invariants can also be evaluated at runtime [9]. This idea is left for a future work.

3.1.2 Running Processes Checker

According to the general approach previously described, the introspector applies introspection to rebuild the list pointed by the *init_task* symbol and retrieve the set of the running processes on a VM. Then, it compares this set against the one returned by executing the command *ps* on the Mon-VM. If the two sets of PIDs differ, then it is very likely that an attacker has replaced critical system binaries with trojaned versions to hide her presence. In case of systems running a fixed number of processes, more severe checks can be defined because the list of allowed PIDs, paired with the name of the processes, can be fixed during the boot of a Mon-VM.

3.1.3 Loaded Modules Authenticator

This function rebuilds the list pointed by the *modules* symbol to retrieve the list of the modules inserted into the kernel. Then, it checks their integrity and that if they are authorized kernel modules. To this purpose, the first time a Mon-VM is started, we load all the kernel modules it can run. For each module, this function computes the hash of the pages storing its code and save these values and the name of the module in a file. Later, when the Mon-VM is running, this function periodically computes the hash of the pages storing the code of each loaded module and checks if this value differs from the previous one or if the name of a module differs from the stored values. Any difference implies that either a module has been modified or a not authorized one has been loaded. An analogous check is applied to the list of open files.

3.1.4 Promiscuous Mode Checker

This function requests the pages starting from the kernel symbol *dev_base*, a pointer to a list of device structures in the Mon-VM. For each structure, the function verifies whether the corresponding flags indicate that the interface is set into promiscuous mode. This approach applies the same checks implemented in [11], but at a distinct level. In fact, while *kstat* accesses these structures at the user-level through */dev/kmem* or, if implemented as a module, at the kernel-level, the introspector function of *Psyco-Virt* applies these checks at the VMM level. Hence, it cannot be defeated even if an attacker gains root privileges.

3.1.5 Anti-spoofing

To support the anti-spoofing capabilities, the IVM Linux kernel is compiled with the following options:

- CONFIG_NETFILTER_XT_MATCH_PHYSDEV
- CONFIG_BRIDGE_NETFILTER
- CONFIG_NETFILTER_NETLINK
- CONFIG_NETFILTER_XTABLES
- CONFIG_BRIDGE

The IVM implements the anti-spoofing checks on the virtual bridge connecting the VMs on the same node using a set of iptables [15] rules. Each rule is defined in terms of the static IP address bound to the virtual interface assigned to a Mon-VM. Every packet with a spoofed source IP address is dropped and logged.

4 Security and Performance Results

This section presents an evaluation of *Psyco-Virt* from the security and the performance points of view. To test *Psyco-Virt* capabilities, we configured an IVM to compute, with a predefined frequency, the hashes of the text area of the kernel and of the modules inside each Mon-VM. In turn, an authorized module into the kernel of each Mon-VM periodically computes the hashes of the text area of critical processes of a Mon-VM, such as the collector, the hosts and network IDses. The IVM also verifies that the list of the loaded modules into the kernel only contains authorized modules, and that the list of running processes and open files does not include hid-

den entries. Lastly, it applies anti-spoofing techniques on the virtual bridge and checks that no Mon-VM is sniffing traffic.

4.1 Effectiveness

To evaluate the effectiveness of the checks implemented by Psycho-Virt, we wrote sample rootkits to update both the text area of a critical process and an entry in the *idt_table* pointing to a modified interrupt handler [18]. We also inserted a malicious module into the kernel of each Mon-VM to modify both a pointer in the *sys_call_table* and an existing system call. Besides, we replaced the system binaries *ps* and *ls* to hide specific processes/files.

The IVM detects the modifications to the system call handler or to an existing system call as well as to pointers in the *sys_call_table* or to an entry in the *idt_table*. The module into the Mon-VM kernel also detects when the text area of a critical process is modified, while the IVM detects each not authorized module loaded into the Mon-VM kernel, as well as a malicious update to an authorized module. Moreover, chkrootkit or Tripwire detects any update to system binaries and the process checker discovers the presence of hidden processes/files. In any of the previous cases, an alert is sent to the director. Finally, the IVM easily detects when an interface is set to promiscuous mode, or when the IP source address of a packet has been spoofed.

4.2 Performance Overhead

To evaluate the overhead due to Psycho-Virt, we ran the IOzone Filesystem benchmark tool [14] on a Mon-VM, while the IVM applies the whole set of consistency checks discussed earlier, with a period between each invocation of 1, 5, 10, 30, 60 seconds or without applying any check. Fig. 2 shows that the maximum overhead on the read test, due to the consistency checks, is about 7%. The same result holds for the write test.

4.3 Limitations

Currently there are some attacks that Psycho-Virt cannot detect. As an example, Psycho-Virt only checks that the *idt_table* and the *sys_call_table* have not been updated, but several other kernel sensitive data structures need to be protected. Moreover, any malicious modification against dynamic data in memory, both in the kernel and user space, is not detected.

As an example, malicious updates of the stack or of the heap are not detected. The complexity of preventing these attacks is very high because a set of memory invariants has to be computed for each process and for the kernel. Another problem arises if an attacker detects the presence of the VMM. In fact, she can try to directly attack the VMM or evade the consistency checks. Evasion is possible provided that sensitive data structures are updated after the checks have been applied and that a consistent state is restored just before the checks are applied again. Finally, provided that an attacker can insert a module the first time the hashes of the authorized modules are computed, a malicious module can be considered an authentic one.

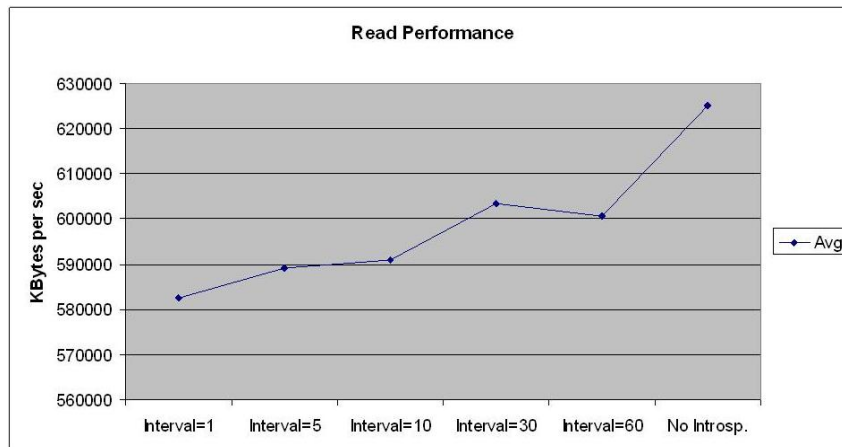


Fig. 2. IOzone Read Performance on a Mon-VM

5 Related Works

VMI is first discussed in [13] together with *Livewire*, which is a prototype of VMI IDS. *ReVirt* [8] supports *recovery*, *checkpoint* and *roll-back* of VMs and uses *virtual-machine replay* to re-execute a system, encapsulated in a VM, instruction-by-instruction for recovering purposes. *IntroVirt* [16] is a system that detects intrusions by executing vulnerability-specific predicates. *Paladin* [1] is a framework that exploits the virtualization technology to detect and contain rootkit attacks. *Manitou* [20] is a system implemented within a VMM that ensures that a VM can only execute authorized code by computing the hash of each page before executing the code and setting the executable bit only if its hash belongs to a list of authorized hashes. [35] describes a hierarchical trust management framework, where

the root of trust is a secure co-processor, which periodically triggers a set of security checkers to build up a chain of trust. The idea of a distributed IDS was first introduced in [29] to monitor a heterogeneous network. The proposed prototype combined data reduction and a centralized data analysis. *Netstat* [32] is a NIDS that applies state transition analysis techniques by modeling intrusions through state transition diagrams. *Hyperspector* [19] is a monitoring environment to detect intrusions in a distributed system, by running each IDS inside a dedicated VM, and connecting all the IDS using an independent virtual network. *Collapsar* [17] is a virtual honeypot architecture to detect network attacks.

6 Conclusion and Future Developments

Psyco-Virt shows that the proposed multi-level approach to achieve highly assurance intrusion detection systems is feasible and effective, and that the overhead is acceptable. The proposed approach defines a three-steps strategy: (i) a set of processes implements HIDS and NIDS tools to detect attacks and intrusions on the Mon-VMs; (ii) specific modules into the kernel of the Mon-VMs check the integrity of these critical processes; (iii) virtual machine introspection is used to protect kernel integrity. In this way, the proposed architecture builds a chain of trust, where each level verifies the integrity of the one above it.

Our future research is focused on the use of introspection to check at runtime a set of memory invariants that should hold for a process or for the kernel. These invariants are computed by applying formal static analysis based on an abstract interpretation approach to the programs or to the kernel code [5, 4, 3]. The set of invariants is an input for the IVM which, for example, may freeze a Mon-VM each time a system call is executed, and apply introspection to check that invariant properties regarding system call parameters are verified.

7 Acknowledgments

We would like to thank Fabio Campisi which helped us in writing and testing the code and the anonymous reviewers for their suggestions.

References

- [1] Arati Baliga, Xiaoxin Chen, and Liviu Iftode. Paladin: Automated detection and containment of rootkit attacks, Jan 2006. Rutgers University Department of Computer Science Technical Report DCS-TR-593.
- [2] R Bradshaw, N Desai, T Freeman, and K. Keahey. A scalable approach to deploying and managing appliances. In TeraGrid 2007, June 2007.
- [3] Thomas Ball, Rupak Majumdar, Todd D Millstein, and Sriram K Rajamani. Automatic predicate abstraction of c programs. In SIGPLAN Conference on Programming Language Design and Implementation, pages 203–213, 2001.
- [4] Francois Bourdoncle. Abstract debugging of higher-order imperative languages. In PLDI '93: Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation, pages 46–55, New York, NY, USA, 1993. ACM Press.
- [5] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fix-points. In POPL, pages 238–252, 1977.
- [6] chkrootkit – locally checks for signs of a rootkit. <http://www.chkrootkit.org/>.
- [7] B. Dragovic, K Fraser, S Hand, T Harris, A Ho, I Pratt, A Warfield, P Barham, and R Neugebauer. Xen and the art of virtualization. In Proceedings of the ACM Symposium on Operating Systems Principles, October 2003.
- [8] 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. In proc. of the 5th symposium on Operating systems design and implementation, pages 211–224, New York, NY, USA, 2002. ACM Press.
- [9] T Fraser. Automatic discovery of integrity constraints in binary kernel modules, Tech. report, University of Maryland Institute for Advanced Computer Studies, December 2004
- [10] The FU rootkit. <http://www.rootkit.com/project.php?id=12>.
- [11] FuSyS. Kstat. http://www.s0ftpj.org/tools/kstat24_v1.1-2.tgz.
- [12] R P Goldberg. Survey of virtual machine research. *IEEE Computer*, 7(6):34–45, 1974.
- [13] T Garfinkel and M Rosenblum. A virtual machine introspection based architecture for intrusion detection. In Proc. Network and Distributed Systems Security Symposium, February 2003.
- [14] IOzone Filesystem Benchmark, <http://www.iozone.org/>.
- [15] Netfilter/Iptables project. www.netfilter.org/.
- [16] Ashlesha Joshi, Samuel T King, George W Dunlap, and Peter M Chen. Detecting past and present intrusions through vulnerability specific predicates. In SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles, pages 91–104, New York, NY, USA, 2005. ACM Press.
- [17] X Jiang and D Xu. Collapsar: A VM-based architecture for network attack detention center. In USENIX Security Symposium, pages 15–28, 2004.
- [18] kad. Handling Interrupt Descriptor Table for fun and profit. *Phrack*, 11(59), July 2002.

- [19] Kenichi Kourai and Shigeru Chiba. HyperSpector: virtual distributed monitoring environments for secure intrusion detection. In VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, pages 197–207, New York, NY, USA, 2005. ACM Press.
- [20] Lionel Litty and David Lie. Manitou: a layer below approach to fighting malware. In ASID '06: Proceedings of the 1st workshop on Architectural and system support for improving software dependability, pages 6–11, New York, NY, USA, 2006. ACM Press.
- [21] P A Loscocco and S D Smalley. Meeting critical security objectives with security enhanced linux. In Proc. of the 2001 Ottawa Linux Symposium, 2001.
- [22] Peter Loscocco and Stephen Smalley. Integrating flexible support for security policies into the linux operating system. In Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, pages 29–42, Berkeley, CA, USA, 2001. USENIX Association.
- [23] Openssl: The open source toolkit for ssl/tls. <http://www.openssl.org/>.
- [24] OpenVPN - An Open Source SSL VPN Solution. <http://openvpn.net/>.
- [25] Open source tripwire. <http://sourceforge.net/projects/tripwire/>.
- [26] Nick L Petroni, Timothy Fraser, Jesus Molina, and William A Arbaugh. Copilot - a coprocessor-based kernel runtime integrity monitor. In USENIX Security Symposium, pages 179–194, 2004.
- [27] S Sparks and J Butler. Shadow Walker: raising the bar for rootkit detection. www.blackhat.com/presentations/bh-jp-05/bh-jp-05-sparks-butler.pdf.
- [28] C Sapuntzakis, D Brumley, R Chandra, N Zeldovich, J Chow, M Lam, and M Rosenblum. Virtual appliances for deploying and maintaining software, 2003.
- [29] Steven R Snapp, James Brentano, Gihan V Dias, Terrance L Goan, L Todd Heberlein, Che lin Ho, Karl N Levitt, Biswanath Mukherjee, Stephen E Smaha, Tim Grance, Daniel M Teal, and Doug Mansur. DIDS (Distributed Intrusion Detection System) - motivation, architecture, and an early prototype. In Proceedings of the 14th National Computer Security Conference, pages 167–176, Washington, DC, 1991.
- [30] sd and devik. Linux on-the-fly kernel patching without LKM. Phrack, 10(58), December 2001.
- [31] Snort - the de facto standard for intrusion detection/prevention. <http://www.snort.org/>.
- [32] Giovanni Vigna and Richard A. Kemmerer. Netstat: A network-based intrusion detection system. Journal of Computer Security, 7(1), 1999.
- [33] VMTN - Virtual Appliance Marketplace. <http://www.vmware.com/vmtn/appliances/>.
- [34] VMware. <http://www.vmware.com/>.
- [35] Lifu Wang and Partha Dasgupta. Kernel and application integrity assurance: Ensuring freedom from rootkits and malware in a computer system. In AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, pages 583–589, Washington, DC, USA, 2007. IEEE Computer Society.
- [36] XenAccess Library. <http://xenaccess.sourceforge.net/>.