Network Based Malware Detection within Virtualised Environments

Pushpinder Kaur Chouhan, Matthew Hagan, Gavin McWilliams, and Sakir Sezer

Centre for Secure Information Technologies, Queens University of Belfast, Northern Ireland, UK

Abstract. While virtualisation can provide many benefits to a networks infrastructure, securing the virtualised environment is a big challenge. The security of a fully virtualised solution is dependent on the security of each of its underlying components, such as the hypervisor, guest operating systems and storage.

This paper presents a single security service running on the hypervisor that could potentially work to provide security service to all virtual machines running on the system. This paper presents a hypervisor hosted framework which performs specialised security tasks for all underlying virtual machines to protect against any malicious attacks by passively analysing the network traffic of VMs. This framework has been implemented using Xen Server and has been evaluated by detecting a Zeus Server setup and infected clients, distributed over a number of virtual machines. This framework is capable of detecting and identifying all infected VMs with no false positive or false negative detection.

1 Introduction

Cloud Computing is a technology which allows consumers access to a broad range of computing resources, products and stored information whenever they need them, where ever they need them, using a variety of devices. Cloud Computing services are marketed as a utility in a similar manner to traditional electricity, gas, water and telephony provision. The simplicity and scalability that cloud computing offers has attracted the attention of both private citizens and enterprises. Virtualisation is the fundamental technology that enables cloud computing and differentiates it from traditional IT deployments by dramatically improving machine utilisation and reducing overall total cost of ownership.

Virtualisation is the emulation of the software and/or hardware platforms upon which other software and operating systems run. Ideally, virtualisation allows us to build an environment that enables one computer to perform the tasks of multiple diverse computing platforms, by sharing the resources of a single hardware platform across multiple virtual systems. An emulated system is called a virtual machine. The operating system installed in a virtual machine is called a guest operating system.

The guest operating systems on a host are managed by either a hypervisor or a Virtual Machine Monitor. This additional software layer controls the flow of instructions between the guest operating systems and the physical hardware e.g. the CPU, disk storage, memory, and network interface cards. A Virtual Machine Monitor (VMM) is a software solution that implements virtualisation in conjunction with or on top of the physical machine's host operating system. In contrast, a hypervisor runs directly on the physical hardware without any intervention from the host operating system. Fig. 1 shows the different level of virtualisation.

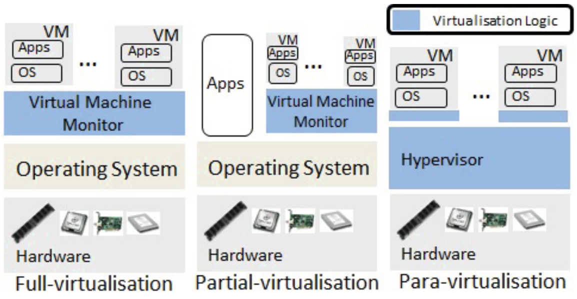

Fig. 1. Level of Virtualisation

In full virtualisation, complete simulation of the actual underlying hardware allows guest operating systems to run unmodified. In partial virtualisation, the virtual machine simulates multiple instances of much of an underlying environment particularly address spaces. In para-virtualisation, the guest operating system needs to be modified to run on top of the hypervisor to access the underlying hardware inside a virtual machine. In hardware assisted virtualisation, the hardware provides architectural support that facilitates building a hypervisor. Hardware-assisted virtualisation enables efficient virtualisation through the use of advanced microprocessors such as Intel VT-x features or AMD-V processor series.

A great advantage of virtualisation is the increase in operational efficiency made by sharing the load of multiple physical systems on a single computer. This provides a potential security benefit by offering a single, centralised platform on which security applications can run. In theory, a single security service running on the hypervisor could potentially work to provide security to all virtual machines running on the system, so long as the virtual machine data is accessible to the hypervisor.

This paper will describe the design and implementation of a hypervisor based framework, which performs specialised security tasks for all underlying guest virtual machines. The aim of this paper is to introduce the following contributions: (1) a new framework for dynamic behaviour-based malware detection in the virtualised environment; (2) a working prototype of this framework; and (3) an evaluation of the proposed framework, validating the feasibility, efficiency and accuracy of its operation.

The rest of the paper is organised as follows. Section 2 describes how the proposed framework relates to and complements the related work in this area. Section 3 explains the framework, its components and mentions the tools and technique used to implemented the framework. Section 4 presents the framework validation results and finally, Section 5 presents the conclusion.

2 Related Work

Malware (malicious software) is a class of software used to disrupt computer operation, gather sensitive information (data and identity theft), or manipulate data within the system (system and data corruption). Malware is any program or file that is harmful to a computer or end-user which is installed without proper consent of the owner. Malware includes computer viruses, worms, Trojan horses, and also spyware, programs that profiles user behaviour.

Researchers are providing new techniques to counter the malware attacks [10,14,4]. One of these approaches is through the use of host based applications to scan the hard disk and memory for known malicious applications or malware traits within executable files. However, malware developers have refined their techniques, with the introduction sophisticated methods such as polymorphism (different encryptions of the same binary) and metamorphism (different code for the same functionality).

Another malware detection technique is to identify the symptoms of malicious behaviour. [10] presented a method that looks for the malware symptoms by using Forensic Virtual Machines (FVMs). FVMs report to a Command & Control module that collects and correlates information so as to take remedial actions in real-time. This method shows effective detection of malware but the main drawback is that an FVM is required for each malicious symptom.

In traditional host based antivirus software, suspicious programs are run in a protected sandbox. If malicious activity is detected then the file can be blacklisted immediately. The sandbox will not operate in a completely isolated network environment and hence there is a small residual risk associated with this technique. Scalability may also cause problems in that it is not possible to provide a sandbox for every device/environment present within an enterprise. To overcome these problems, AV vendors have started to leverage the cloud to track the reputation of individual files.

CloudAV [14] is a program that provides antivirus protection as a cloud service. CloudAV allows the user to take advantage of multiple antivirus programs without running them locally so the user's computer performance is not affected. The program uses a technique called N-version protection to identify malicious software by using multiple, heterogeneous antivirus detection engines in parallel. However, file hashes and byte (or intrusion) signatures can be obfuscated through the use of polymorphism. In addition, the CloudAV is reliant on signature based antivirus products which may not detect new m alware quickly enough. The latency window of exposure (when attack occurs and when specific malware files are known) has to be taken into account.

Some researchers have proposed a twin track solution to protecting the privacy of Cloud Service user data. By separating trusted computing on to one cloud service and passing all other data to a second cloud service. More generally they propose clouds for separate computation of arbitrary functions according to different security and performance criteria. TwinClouds [4], by using two clouds, raises redundancy and privacy issues relating to the use of shared storage mechanisms.

Terra [7] is a VM-based architecture for supporting various security models on a single physical machine by combining the good aspects of closed and open box platforms. In closed box platform service provider has control over applications, content, and media, and restricts convenient access to non-approved applications or content. Where as in open platform consumers have unrestricted access to applications, content, and much more. Terra considers the implementation of a trusted virtual machine monitor on top of trusted platform module. The Trusted Virtual Machine Monitor verifies that hosts are trusted by the cloud service user. Terra prevents the owner (Cloud service provider) of a physical host from inspecting and interfering with the computation of a running VM in a trusted host.

Trusted Cloud Computing Platform (TCCP) provides a closed box execution environment which makes the computation taking place in a virtual machine confidential to privileged users. Thus, if TCCP [16] is deployed by a service provider such as Amazon EC2, cloud service users can verify that computation is confidential. Even a privileged user with access to the VM state cannot obtain user data. The main limitation of TCCP is that every virtual machine has to be launched on a trusted VMM which may reduce the elasticity feature of cloud computing.

Although frameworks such as Terra [7], TCP [17], TCCP [16], and improved-TCCP [9], allow users to attest whether the service is secure and/or running on a secure host before the launch of virtual machine, however, after the launch of a virtual machine an attack by an external bot has to be determined by other means. Thus, our proposed framework is complementary to existing trusted cloud computing frameworks.

ReVirt [5] is a virtual machine based logging and replay system that attempts to address the lack of completeness provided by traditional system loggers. ReVirt logs instruction level detail for each virtual machine and then carries out analysis looking for malicious activity. However, the main drawback of this system is that it too may leak sensitive information if the logs are not well protected. Our framework overcomes this drawback, as all the sensitive and malware related information is stored on a separate secure system.

Many other VMM based security systems (LiveWire [8], Siren [3], SubVirt [12], VM-based IDS [19,1]) have been developed based on VM-isolation, VMM-inspection and VMM-interposition capabilities. However, to construct a truly effective and efficient virtual machine monitor based security system the functionality of the previously mentioned detect malicious activities in real-time. Our proposed framework features network (VM communication) based malware detection in real time. The network traffic analysis is useful in malware detection in virtualised environments as it allows us to observe a wide range of behaviour e.g. browser requests, sending an email or a file transfer. The attacker may use the victims network connection to perform malicious activities, such as participate in DDoS attacks, connect to a malicious command and control server, or other attacks. The network activity may also consist of data exfiltration, whereby confidential information may be extracted from the machine, through various automated or manual methods, and transmitted to the attacker.

In virtualised environments all virtual machine network traffic must pass through the hypervisor. Hence there is an opportunity for the hypervisor to passively observe this network activity to perform malware detection. It is envisaged that upon the detection of malware activity, the hypervisor can log activity or interact with the infected virtual machine by shutting it down or providing a warning message to the user. The concept of network based analysis to detect malware behaviour through network analysis is complementary to host-based malware detection techniques (signature-based, symptoms-based, etc.) used for Virtual Environment protection.

3 Malware Detection System for Virtual Environments

The Malware Detection System for Virtual Environments (MDSVE) observes network traffic and identifies any patterns and trends that indicate activities which can potentially have malicious effects on the virtualised environment. The overall functionality of MDSVE is to capture the network traffic of each virtual machine and build useful contextual information to aide traffic analysis. Detected threats will precipitate security alerts or direct action on the VMs involved. Tasks performed by MDSVE are:

1. Track virtual machine lifecycle
2. Monitor virtual machine communications i.e. internal communications between VMs in the same physical host, and external communications traversing a network interface card on the physical host
3. Capture malware activities
4. Match any suspicious activities with the corresponding virtual machine.
5. Inform management console about suspicious virtual machine instances.

3.1 Components of MDSVE

The MDSVE is made up of four basic functional blocks: network sniffer, malware trait detector, virtual machine information collector, and virtualisation security manager, as shown in Fig. 2.

Network Sniffer- (NS) captures all of the network packets in real-time and performs flow classification. That is, all packets relating to a logical session are linked together and offered up for further analysis as a contiguous flow of traffic.

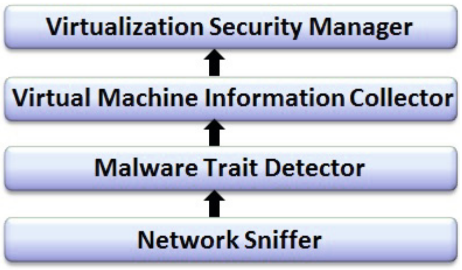
Virtualization Security Manager
Virtual Machine Information Collector
Malware Trait Detector
Network Sniffer

Fig. 2. Architecture of Malware Detection System for Virtual Environment

A PCAP library based platform called ITACA (Internet Content and Traffic analysis) [11] is used for this purpose.

Malware Trait Detector- (MTD), analyses the complete flow looking for series of events and features which indicate the presence of specific malware. Some malware detection techniques and tools make use of network analysis; for example, Bothunter [15] attempts to identify generic traits within network traffic, such as malware downloaded.

Virtual Machine Information Collector- (VMIC) monitors the virtual machine life cycle and captures the virtual machine status along with basic parameters so as to match the virtual machine with the suspicious bot as analysed by the malware analyser and alerts the virtualisation security manager.

Virtualisation Security Manager- (VSM) acts as a security console displaying the alerts and information provided by the VMIC. If malware is detected on any of the virtual machines, then according to the severity of the threat VSM can take action accordingly. For example, inform host based anti-virus system, sinkhole traffic destined to the virtual machine, or even suspend the virtual machine.

3.2 Setup Malware Detection System for Virtual Environment

Based on our framework, a prototype of MDSVE was implemented using some pre-existing software and tools: Xen is used for the virtual environment setup, ITACA [11] is used to capture the network traffic, the Zeus [6] botnet server is used as the primary source of malware infection, and a Zeus malware detection plugin for ITACA [6] provides a malware trait detector. A new software component implements the VMIC functionality.

Xen Virtualised Environment- Xen [2] is used to build virtualised environment because virtualisation over head remain under 3% for virtualising Linux, FreeBSD and Windows XP. Xen [13] is an open-source native, type-1 or bare-metal hypervisor. A Xen guest typically has access to one or more paravirtualised (PV) network interfaces. These PV interfaces enable fast and efficient network communications for domains without the overhead of emulating a real network device.

A paravirtualised network device consists of a pair of network devices. The frontend will reside in the guest domain while the backend will reside in the backend domain (typically Dom0). A similar pair of devices is created for each virtual network interface. The front and backend devices are linked by a virtual communication channel. Frontend devices generate the traffic that has to be transported. Guest networking is achieved by arranging for traffic to pass from the backend device onto the wider network, using bridging, Network Address Translation or routing.

Network Sniffer- Internet Traffic And Content Analysis (ITACA) [11] is a network packet sniffer that monitors network traffic in real-time, scrutinizing each packet closely to detect a dangerous payload or suspicious anomalies, developed at CSIT, Queen's University of Belfast. ITACA is based on libpcap, a tool that is widely used in TCP/IP traffic sniffers and analysers. ITACA captures packet traffic and builds derived sets of data, from which the correctness of protocol formats are established. The ITACA platform enables the creation of sophisticated security analysis systems using modular plugin functions implemented in software and/or hardware.

ITACA has a three layer architecture; the network layer, the ITACA core and the plugin layer. The network layer interfaces with the network to extract the raw bytes of packet data using PCAP library. These captured bytes are passed to the ITACA core to extract and process all available information such as the 5-tuple (Source and Destination IP Address, Source and Destination Port Number and Protocol ID) which characterises a flow or logical session. The plugin layer is used to support multiple customised traffic treatments that operate independently and efficiently.

Plugins are created using a well-defined C++ API and make use of an event driven architecture optimised for multi-threaded operation.

ITACA is used to perform the task of two components in the MDSVE architecture; the network sniffer and the malware trait detector. The network layer and the ITACA core layer provide network sniffer functionality whereas the plugin layer provides the malware trait detector.

Malware Trait Detector- (MTD) is implemented by a malware detection plugin in ITACA. The Plugin layer of ITACA allows the implementation of specialist network analysis methods. During registration with ITACA, plugins specify the types of traffic that are of interest to them. Plugins operate in parallel to the ITACA core (and each other), allowing the running of additional plugins with limited effect on system performance.

For the prototype implementation of our framework, a Zeus detection plugin was deployed. Zeus was selected for testing because of its predominance it is the most popular botnet amongst online criminals, with a prevalence rate against other botnet software of 57.9% according to a recent McAfee study [18] which analysed half a million malware samples from January to March 2013.

The function of the Zeus botnet plugin is to analyse traffic and detect the periodic communication which takes place between an infected machine and the Zeus command and control servers. Rather than detecting the infection mechanism employed, which may include methods like browser exploitation or social engineering, the plugin only aims to detect malware network traffic activity, subsequent to infection.

Virtual Machine Information Collector- (VMIC) detects if any of the running VMs on a host are infected with malware. The function of VMIC is to correlate reports of malicious network activity (from the Zeus detector plugin) with the VM status tables i.e. to match the network activity with a running VM. The VMIC then inform the management console (VSM).

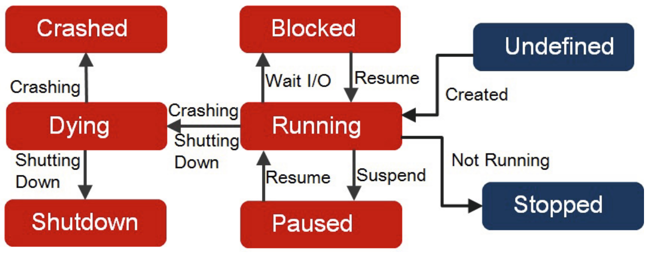

Fig. 3. Xen VM Lifecycle-flows from one state to another

Xen domain can be in one of the six states (shown in Fig. 3). A virtual machines state can be displayed in VMM or by viewing the results of the xm list command, which abbreviates the state using a single character.

r - running - The virtual machine is currently running and consuming allocated resources.
b - blocked - The virtual machines processor is not running and not able to run. It is either waiting for I/O or has stopped working.
p - paused - The virtual machine is paused. It does not interact with the hypervisor but still maintains its allocated resources, such as memory and semaphores.
s - shutdown - The guest operating system is in the process of being shutdown, rebooted, or suspended, and the virtual machine is being stopped.
c - crashed - The virtual machine has crashed and is not running.
d - dying - The virtual machine is in the process of shutting down or crashing.

Detailed information of each VM is collected and stored as a table of records which include the VMID, VMName, VM MAC address, installed OS on the VM, state and start time of the VM.

To find the malware infected virtual machine, the VMIC matches the MAC address of the virtual machine with the source or destination MAC address of the malicious network packet.

The virtual machine information table is updated dynamically as and when VM status changes occur. Reported Zeus botnet features are collected and evaluated for each VM instance. A threat index is calculated and when this exceeds a high-water mark the virtual machine is deemed to be malicious and the VSM is informed immediately.

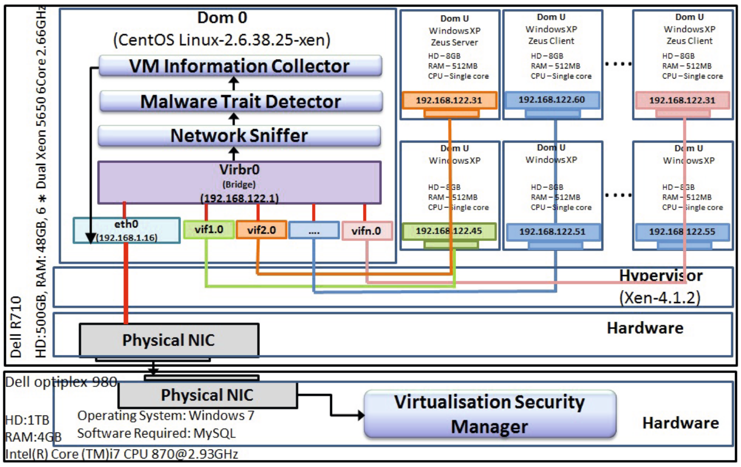

Fig. 4. Network Based Malware Detection System for Virtualised Environment

4 Validation of the Malware Detection System within a Virtualised Environment

The objective of this experiment is to validate that the proposed network-based malware detection framework is a viable solution for future cloud and application security. The validation requires that a small subset of virtual machines are infected with known malware creating a realistic scenario. It is also assumed that the virtual machines produce traffic patterns typical to virtualised environments. This section presents the experimental design, procedure and results to validate the proposed framework.

4.1 Experimental Setup

To evaluate the proposed framework, a prototype implementation was deployed within the hypervisor. The implementation consisted of the three essential components of the architecture: the network traffic analyser, the malware trait detector and the virtual machine information collector. A fourth component, the virtualisation security manager, was deployed on a separate machine acting as a security console. All four components run as independent processes. An SQLite database was used by the VMIC to store the VM information. The test-bed is shown in Fig. 4 with connectivity between a few VMs depicted.

Xens default network configuration was used, bridging within the backend domain (Dom-0). This allows all domains to appear on the network as individual systems. ITACA analyses all traffic traversing the default gateway (e.g. eth0) and any intra-VM communications on the virtual bridge.

Table 1. Validation of Malware Detection System For Virtualised Environment

VM Name	Infected	Detected	Malware Feature detected- at Time	Correct
Zeus Client1	√	√	External Connection-12:22:15 Beaconing Pattern-12:39:20	√
Zeus Client2	√	√	External Connection-12:22:15 Beaconing Pattern-12:39:24	√
Zeus Client3	√	√	External Connection-12:22:17 Beaconing Pattern-12:39:28	√
Zeus Client4	√	√	External Connection-12:22:17 Beaconing Pattern-12:39:31	√
Clean1	X	X		√
Clean2	X	X		√
Clean3	X	X		√

4.2 Network Traffic Capturing

To validate the proposed framework two types of virtual machines were launched in the testbed environment. 5 VMs are used to form a Zeus botnet (1 Zeus Server and 4 Zeus Clients) and 3 VMs are benign, which generate random traffic by running the web access applications (email, Dropbox, Facebook and Skype).

The network packets captured from traffic between VMs (VM-VM communication) and between VM and outside Virtual environment (VM-external machine communication) are analysed by the framework.

4.3 Experimental Result

Under default settings, firm detection of Zeus takes one hour, usually with a short additional time period to account for network connection latency. The default Zeus configuration beaconing period is 60 minutes with the shorter exfiltration event running every 20 minutes. The result of Zeus network traffic detection experiment is shown in Table 1. Infected VMs were detected by the proposed framework based on the Zeus detection features. The time delta between infection and Zeus feature detection demonstrates malware detection within reasonable time, assuming use of the default Zeus configuration. All infected VMs are detected, without false positives reported. These experiment results demonstrate correctness of the proposed framework.

4.4 Privacy and Security Issues

One area of concern is the fact that all network traffic captured is observed by software for the purposes of detecting malicious activity. This may raise privacy concerns with network operators and users, as network monitoring may be used for gaining information about network users. However, within this framework, network monitoring software will be run with the sole aim of detecting malicious activity, with only the VMID and IP addresses being reported during a suspicious event. While a false positive result may inadvertently disclose a connected IP address, the program will reveal no further information to an administrator. In terms of information observed, the detection system is no more intrusive than other commonly used IDS.

In terms of security, having a powerful monitoring entity on the hypervisor is potentially a concern, should the hypervisor be compromised. However, in order to attack users on the network, the attacker would need to install and utilise their own detection utility. The presence of the malware detection utility would have little relevance within this attack, since the attacker would merely be able to use the application as intended by the administrator or disable it, rather than use it maliciously.

5 Conclusion

In this paper, a new network based malware detection framework for virtualised environments has been proposed and experimentally proven. The proposed framework is advantageous in a number of ways. For example, the proposed system is scalable in that it can function with a high number of users and traffic while remaining functional. In terms of performance, under normal conditions, the overhead of deploying the system is negligible as only one additional application is needed on the hypervisor to detect malicious activity across all Virtual Machines. The framework is accurate, in that it divises a method of uniquely identifying a virtual machine based on its MAC address.

The proposed framework detected all the malware infected VMs without false positive or false negative detection of Zeus bot. This paper has shown that the work done by the forensic community in malware detection through network analysis is directly applicable to Virtual Environment malware detection. By providing interfaces between the two worlds, the difficulty of developing new virtual security solutions can be significantly reduced. It is envisaged that this work can be used as a basis for virtual machine security, in that a centralised hypervisor process can perform security related detection and scanning functions for all the virtual machines it is hosting. Such processes would enable greater convenience and security for the end user of the virtual machine, as well as decreasing security based application and management overhead.

References

1. Azmandian, F., Moffie, M., Alshawabkeh, M., Dy, J., Aslam, J., Kaeli, D.: Virtual machine monitor-based lightweight intrusion detection. SIGOPS Oper. Syst. Rev. 45(2), 38–53 (2011)
2. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. SIGOPS Oper. Syst. Rev. 37(5), 164–177 (2003)

3. Borders, K., Zhao, X., Prakash, A.: Siren: Catching evasive malware (short paper). In: Proceedings of the 2006 IEEE Symposium on Security and Privacy, SP 2006, pp. 78–85. IEEE Computer Society, Washington, DC (2006)
4. Bugiel, S., Nürnberger, S., Sadeghi, A.-R., Schneider, T.: Twin clouds: Secure cloud computing with low latency. In: De Decker, B., Lapon, J., Naessens, V., Uhl, A. (eds.) CMS 2011. LNCS, vol. 7025, pp. 32–44. Springer, Heidelberg (2011)
5. Dunlap, G.W., King, S.T., Cinar, S., Basrai, M.A., Chen, P.M.: Revirt: Enabling intrusion analysis through virtual-machine logging and replay. SIGOPS Oper. Syst. Rev. 36(SI), 211–224 (2002)
6. Falliere, N., Chien, E.: Zeus: King of the bots (2009)
7. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: A virtual machine-based platform for trusted computing. In: 9th ACM Symposium on Operating Systems Principles, SOSP 2003, pp. 193–206. ACM, New York (2003)
8. Garfinkel, T., Rosenblum, M.: A virtual machine introspection based architecture for intrusion detection. In: Proc. Network and Distributed Systems Security Symposium, pp. 191–206 (2003)
9. Han-zhang, W., Liu-sheng, H.: An improved trusted cloud computing platform model based on daa and privacy ca scheme. In: 2010 International Conference on Computer Application and System Modeling (ICCASM), Oct 2010, vol. 13 (2010)
10. Harrison, K., Bordbar, B., Ali, S.T.T., Dalton, C.I., Norman, A.: A Framework for Detecting Malware in Cloud by Identifying Symptoms, pp. 164–172. IEEE (2012)
11. Hurley, J., Munoz, A., Sezer, S.: Itaca: Flexible, scalable network analysis. In: ICC, pp. 1069–1073. IEEE (2012)
12. King, S.T., Chen, P.M., Wang, Y.-M., Verbowski, C., Wang, H.J., Lorch, J.R.: Subvirt: Implementing malware with virtual machines. In: IEEE Symposium on Security and Privacy, SP 2006, pp. 314–327. IEEE Computer Society (2006)
13. Nguyen, A.-Q., Takefuji, Y.: A novel approach for a file-system integrity monitor tool of xen virtual machine. In: Bao, F., Miller, S. (eds.) ASIACCS, ACM (2007)
14. Oberheide, J., Veeraraghavan, K., Cooke, E., Flinn, J., Jahanian, F.: Virtualized in-cloud security services for mobile devices. In: 1st Workshop on Virtualization in Mobile Computing, MobiVirt 2008, pp. 31–35. ACM, New York (2008)
15. Porras, P.A.: Directions in network-based security monitoring. IEEE Security & Privacy 7(1), 82–85 (2009)
16. Santos, N., Gummadi, K.P., Rodrigues, R.: Towards trusted cloud computing. In: Proceedings of the 2009 Conference on Hot Topics in Cloud Computing, HotCloud 2009. USENIX Association, Berkeley (2009)
17. Shen, Z., Li, L., Yan, F., Wu, X.: Cloud computing system based on trusted computing platform. In: International Conference on Intelligent Computation Technology and Automation, ICICTA 2010, vol. 01. IEEE Computer Society (2010)
18. Thakar, N.: Botnets remain a leading threat (2013), `https://blogs.mcafee.com/business/security-connected/tackling-the-botnet-threat`
19. Wang, H., Zhou, H., Wang, C.: Virtual machine-based intrusion detection system framework in cloud computing environment. JCP 7(10), 2397–2403 (2012)