Integrated Security Architecture for Virtual Machines

Vijay Varadharajan and Udaya Tupakula

Information and Networked Systems Security Research
Faculty of Science, Macquarie University, Sydney, Australia
{vijay.varadharajan,udaya.tupakula}mq.edu.au

Abstract. Currently virtualisation technology is being deployed widely and there is an increasing interest on virtualisation based security techniques. There is a need for securing the life cycle of the virtual machine based systems. In this paper, we propose an integrated security architecture that combines access control, intrusion detection and trust management. We demonstrate how this integrated security architecture can be used to secure the life cycle of virtual machines including dynamic hosting and allocation of resources as well as migration of virtual machines across different physical servers. We discuss the implementation aspects of the proposed architecture and show how the architecture can counteract attack scenarios involving malicious users exploiting vulnerabilities to achieve privilege escalation and then using the compromised machines to generate further attacks.

Keywords: Virtualisation, Trusted computing, Access Control, Intrusion detection, Security attacks.

1 Introduction

Security issues play a vital role in every organisation, as greater availability and access to information in turn imply that there is a greater need to protect them. To address this issue, several access control mechanisms, languages and systems [1-6] have been proposed in the past. Many of these systems make certain basic assumptions about the state of the platform that is hosting and running the applications and systems software. There is an inherent trust that is placed on the underlying platform when a user or an upper level application is authenticated or authorised to perform actions. In the current networked world with heterogeneous platforms and numerous software applications and system software running on these platforms, it is important such underlying trust assumption about the system state be properly examined. There are several reasons for this. Firstly, computing platforms have become very powerful and can run many applications simultaneously. In particular, as the number of software applications increases, greater is the possibility for security vulnerabilities. These vulnerabilities in turn make the platform more vulnerable to attacks. Secondly, attacks themselves are becoming more and more sophisticated. Furthermore, attackers also have easier access to ready-made tools that enable exploitation of platform vulnerabilities more effective. Thirdly, platforms are

T. Zia et al. (Eds.): SecureComm 2013, LNICST 127, pp. 140–153, 2013.

being shared by multiple users and applications (belonging to different users) both simultaneously as well as at different times. Therefore there is a great chance of the platform being left in a vulnerable state as different users and applications run. Finally, because platforms have become much more complex today, users themselves are unaware of their platform vulnerabilities. Hence there is need for techniques for integrated security techniques for enhancing the security of the systems.

In this paper, we propose an integrated security architecture which combines policy based access control intrusion detection techniques and trusted computing for securing the lifecycle of distributed applications running on virtual machines. The paper is organized as follows. Section 2 presents an application scenario which highlights the need for such integrated security techniques in the current environment. In Section 3, we propose novel integrated security architecture for securing the life cycle of virtual machine based distributed applications. Section 4 presents the implementation details. Finally, Section 5 concludes the paper.

2 Application Scenario

The current networked environment is characterised by different types of security attacks and the attacks dynamically changing to avoid detection and prevention. Given the heterogeneous nature of the technology spectrum with different operating system platforms, fixed and mobile, with different networks and numerous different applications, the range of attacks possible is wide ranging. Hence it is complex and difficult to detect and prevent these different types of attacks using single security technologies such as access control or intrusion detection and prevention. There is a need to develop integrated security architecture combining different security functionalities as well as deploy a range of security tools such as access control mechanisms and intrusion detection systems. Such an integrated architecture is necessary to deal with emerging attacks. Let us consider an example scenario which illustrates the need for integrated security architecture.

Consider the scenario in Figure 1, where we have distributed system architecture with applications running on virtual machines (VMs) on top of a Virtual Machine Monitors (VMMs) [7]. Let us assume that a customer requests to host virtual machines in this distributed datacentre architecture. Consider the case where the VMMs have security tools such as access control and intrusion detection to protect their resources from security attacks. Each VMM may have its own access control policies for hosting virtual machines. For instance, assume that Chinese wall access policy [8] is being enforced by the access control system as shown in Figure 1. Assume that there are VMs hosting requests from banks and oil companies. With the Chinese wall policy, if Bank A's VM is hosted on VMM1, then say Bank B's VM cannot be hosted on the same VMM1. However it can be deployed on VMM2. Now consider a Bank C's VM which cannot be hosted on any of the VMMs. If Bank C's VM is hosted on VMM1, then there is a possibility for information leakage between Bank A's VM and Bank C's VM. Similarly if Bank C's VM is instantiated on VMM2, then there could be information leakage between Bank B's VM and Bank C's

VM. Hence the security architecture should not allow Bank C's VM to be deployed on any of the VMMs. However, it will allow virtual machines belonging to an oil company to be hosted on the VMM. Either the datacentre administrator has to deploy a new physical server to host the Bank C's VM or host the Bank C's VM only when one of the other Banks' VM terminates or shuts down.

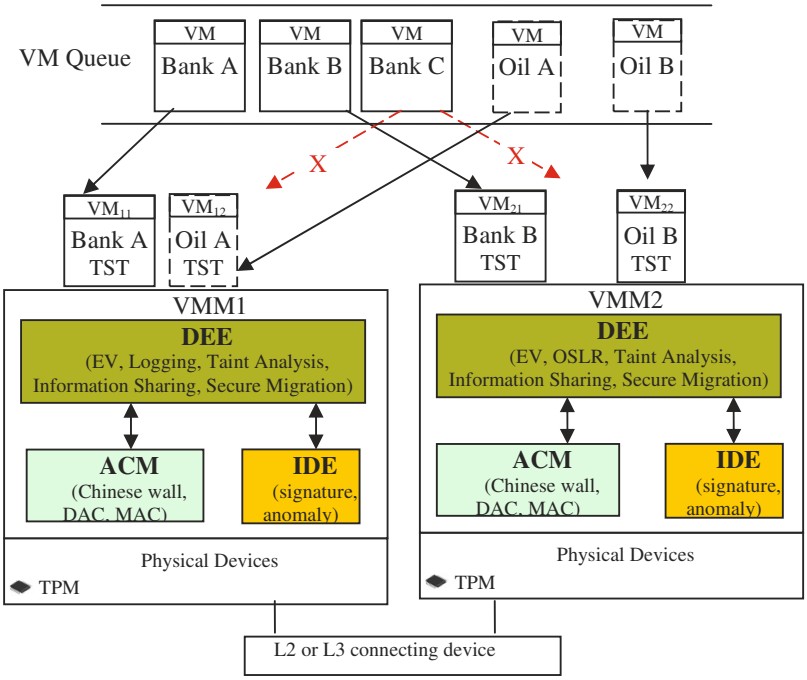

Fig. 1. Integrated Security Architecture for Virtual Machines

In addition, there is also a need to ensure secure operation of the virtual machines. Note that the intrusion detection tools have been deployed at the VMM instead of the VMs to ensure that they themselves do not succumb to attacks at the VMs (which are the ones that are being monitored). Hence the intrusion detection security tool at the VMM should detect if an attacker exploits some known vulnerabilities in the VM (or any traditional security tools (TST) such as [6] that are present at the VM) to generate attacks. Furthermore, in the distributed environment, there may be a need for migration of virtual machines to different physical machines. That is, a VM1 running on top of VMM1 may migrate to a VMM2. Hence the security architecture needs to ensure that the virtual machine can be migrated in a secure manner. Hence we can see the need for a security architecture that brings together access control policies and

mechanisms with the intrusion detection and prevention mechanisms in a trusted manner to respond to dynamic changes in attacks.

2.1 Integrated Security Approach

Let us now consider logical security functionalities that need to be combined in the integrated security architecture.

There have been numerous security models that focus on access controls. We mentioned the Chinese wall policy in the previous section above when describing the application scenario. The traditional access control models include discretionary access control models such as those based on access control lists, mandatory access control models such as those based on security labels, type enforcement models, information flow models as well as role based access control models. In principle, each of these can be applied in a virtual machine based distributed systems context. For instance, sHype architecture [9] addressed the enforcement of mandatory access control for virtual machine based systems. It provides a reference monitor interface inside the hypervisor (VMM) to enforce information flow constraints between virtual machine partitions. When a virtual machine partition makes a request to access a shared virtual resource, an access control module in the VMM acting as the reference monitor enforces the mandatory access control policy. Extending this to a distributed system, a distributed application can be represented as a collection of virtual machines that execute across different physical machines. Using such a system, we can achieve a range of access policies such as type enforcement, Bell-LaPadula [10], Chinese wall as well as information flow type security policies.

Access control systems are concerned about preventing access to the resources by the unauthorised users. However if a user (attacker) is successful in obtaining high level privileges through any means such as exploiting a vulnerability such as buffer overflow or by using stolen credentials, access control systems will not be able to differentiate these (malicious) users and are not able to enforce any restrictions on the actions performed by them. Hence by gaining unlawfully the higher level privileges, the attackers are successful in performing malicious activities such as installing malicious software or altering the legitimate applications and using these compromised systems to generate attacks. Such attacks are often detected by the intrusion detection systems in the traditional environment since they have signatures or baseline behaviour for the normal use of the systems or entities. In this case, although the attacker has obtained higher privileges, the actions performed by the malicious users (such as installing root kits and altering ls code to hide the malicious process) either match with the signatures stored in the attack signature database or deviate from the normal behaviour of the system. Hence the intrusion detection system raises an alarm when suspicious activity is detected.

Integrating the intrusion detection mechanisms in the VMM gives rise to several advantages. It provides isolation as the VMM itself is protected from the vulnerabilities in the applications and operating system in the guest virtual machines. Also as the VMM has control on the resources of the system such as memory and I/O devices, it is able to inspect the resources allocated to virtual machines. Hence the

intrusion detection mechanisms if they are placed in the VMM can take advantage of this capability in their evaluations. Furthermore, as the intrusion detection code is interposed between a malicious virtual machine and the attacked resource, this interposition enables efficient detection of attacks. For instance, Dunlap et al [11] proposed ReVirt architecture for secure logging by placing the logging tool inside the VMM. Garfinkel [12] proposed a Livewire intrusion detection system which makes use of the VMM to achieve introspection and obtain the state of the virtual machines. However in a distributed environment, we need to be able to detect attacks not only from a single VMM but from distributed VMMs by sharing information about intrusions between them in a secure manner. Lycosid [13] detects hidden process in the virtual machines by comparing the implicit guest view with the VMM image. If the number of processes reported by the guest VM does not match with the number of processes identified by the VMM then there is a hidden process. It does not address attacks generated by visible process.

However there are also some additional challenges with the intrusion detection systems. For example, signature based systems cannot deal with the zero day attacks and anomaly based tools have higher false alarms. Our observations confirm that many attacks first exploit one or more weaknesses in access control followed by the malicious activity. However since the access control and intrusion are often implemented separately, they are not efficient in detecting and preventing sophisticated attacks. Furthermore, traditionally access control systems have been implemented in the operating system or at the applications by the respective vendors, and intrusion detection systems are installed and configured by the end users. Hence there is a need for integrating the access control and intrusion detection tools in a virtualised environment for greater efficient detection of attacks. Such an integrated security architecture should also address additional challenges that arise in a virtual environment such as dynamic hosting of virtual machines on the VMM, dynamic varying of the allocated resources and migration of the virtual machines between different physical servers. Hence the architecture should support techniques that can ensure secure hosting, secure operation, and secure migration of the virtual machines.

Finally let us consider the integrity and trustworthiness of the VMM platform itself. The third logical functionality that we would like to consider in the integrated security approach is that of trust management, which helps to establish the trust on the VMM platform. The notion of trust is the expectation that an entity will behave in a particular manner for a specific purpose. A trusted platform is a platform that contains hardware based subsystem and special processes (Trusted Platform Module (TPM) [14]), which dynamically collect and provide evidence of behaviour. These special processes themselves are “trusted” to collect evidence properly. There are also third parties endorsing platforms which underlie the confidence that the platform can be “trusted”. The basic idea is if the physical machine has the TPM, then using its mechanisms, one can measure the state of the VMM on boot and confirm that the VMM is brought into a trustworthy state, if it matches with some reference state. Once the VMM with its access control and intrusion detection functionalities are in a trustworthy state, then the guest virtual machines can be loaded onto the secure VMM.

This completes our integrated security architecture which brings together the access control, intrusion detection and trust management functionalities into the virtual machine based distributed system environment.

3 Security Architecture Overview

Consider the secure and trusted VMM based architecture diagram shown in Figure 1, where each physical server is equipped with a hardware trusted platform module (TPM) chip. Within the VMM, security functionalities of access control, intrusion detection and security decision evaluation have been implemented using the modules Access Control Module (ACM), Intrusion Detection Engine (IDE) and Decision Evaluation Engine (DEE) respectively. This architecture is used to manage the security life cycle of a virtual machine such as secure hosting of a virtual machine, its secure operation as well as secure migration of virtual machines.

There are several components to the DEE module that perform entity validation, logging, taint analysis, information sharing, and secure migration. The entity validation in the DEE is responsible for determining the entity at fine granular level. Note that the entity can vary depending on the action associated with the virtual machine. For example, before hosting a VM on the VMM, the VM is considered as an entity. After the VM is hosted, the processes running in the VM can be considered as entities. After the entity is determined by the entity validation component, the DEE makes a decision on the entity by considering the security policies in the ACM and the IDE components. The ACM module is used for enforcing different access control policies such as Chinese wall to prevent conflicting virtual machines being hosted on the same server; e.g. virtual machines from Bank A and Bank B. Furthermore, it also has techniques to detect privilege escalation attacks performed by the users in the virtual machines. The IDE module is used for detection of both known attacks as well as suspicious behavior by monitoring the interactions of virtual machines. Some components such as entity validation and logging are active for all events on the virtual machines whereas other components such as taint analysis [15] are invoked for specific actions on virtual machines. Taint analysis is invoked only when the ACM or IDE detects some suspicious activity in the virtual machine. Information sharing component is only used for sharing attack information between different secure VMM based physical servers in the distributed environment. Secure migration component is used to validate the capability of the remote physical server to which the virtual machine will be migrated.

Now let us consider how these modules in our secure and trusted VMM based architecture can be used for securing the life cycle of the virtual machines.

3.1 Secure Hosting

Most of the current virtualisation systems support dynamic hosting of virtual machines. However there is a need to ensure secure hosting of the virtual machines. The DEE module in our model is concerned with ensuring secure hosting of virtual

machines on the VMM. Before hosting a virtual machine on the VMM, the entity validation component in the DEE module determines whether the virtual machine conflicts with any of the access control policies enforced in the ACM module. If any of the conflicting virtual machines are already running on the VMM, then the DEE module prevents hosting of the virtual machine on the VMM. If hosting of the virtual machine does not conflict with any of the running virtual machines, then the DEE evaluates other factors such as the available resources. If the DEE then decides to host the virtual machine, then the TPM based system is used to measure the state of the virtual machine and ensures it is trustworthy at boot time.

Now let us consider briefly how TPM based system can be used to ensure that VM boots into secure state for completeness. A Trusted Platform includes a Trusted Platform Module (TPM) chip, a Core Root of Trust for Measurement (CRTM), TCG Software Stack (TSS) and the related certification. The TPM is a hardware chip that performs cryptographic functions and is separate from the main CPU. The CRTM is the first piece of software to run as the platform is booted. The TSS is the software code that is needed to perform various functions of the Trusted Platform. There are also a number of Certification Authorities that issue a certificate vouching that various features of the Trusted Platform are genuine. Once the platform is booted, the CRTM measures itself to ensure that it has not been compromised and stores the measured value in the Platform Configuration Register (PCR) of the Trusted Platform Module. For this reason, the TPM is also called as the Root of Trust for Storage (RTS). Then, the CRTM passes control to the first measurement agent (MA). A bootstrapping process follows where all agents measure the software modules they are responsible for and store the measured values inside the PCRs. The process continues until the last measurement agent has recorded the value inside the TPM. This way, at every boot, the TPM stores the measurement values of all the software components of the Trusted Platform. This ensures that the VMM, the VM operating system and its applications are in secure state during boot time.

3.2 Secure Operation

Now let us consider how the DEE ensures secure operation of the virtual machines.

The entity validation and the logging in the DEE are invoked for all the actions on the virtual machines. The entity validation component identifies the entities at fine granular level and determines which security policies in the ACM or IDE component need to be enforced on the interactions of the entity. Logging is used for capturing the specific features of the virtual machine and the entity interactions. In addition to the security policies in the ACM and the IDE which are able to detect the attacks, the DEE determines whether additional policies need to be enforced on the VM entities. Whenever suspicious behaviour is identified by the ACM or the IDE components, the DEE can decide to perform taint analysis to determine if the suspicious behaviour is actually malicious. The DEE can also be used for sharing of information between the different secure VMM physical servers. For instance, this can happen when new attacks are discovered by one physical server and shared with others.

Now let us consider a common attack scenario involving privilege escalation during the operation of the system and see how this attack can be detected with the proposed integrated security architecture. For example, consider a user who has logged on with limited privileges exploits some vulnerability to obtain higher privileges and performs malicious activities such as disabling security tools that have been installed in a virtual machine or installs some malicious programs in the virtual machine.

First let us consider the privilege escalation by exploiting vulnerabilities in the SQL server. In one of the attacks mentioned in [16], a user who has logged in with limited privileges obtains administrative privileges by changing three bytes in the memory by exploiting buffer overflow vulnerability. SQL server validates the user id before giving access to any of the objects. If the user id is set to 1, then the user is considered to have the administrative privileges. The user can alter the id in the memory in the vulnerable server after calling VirtualProtect(). The administrative privileges of such malicious users will be valid until the SQL server is restarted. Hence a malicious user can use such temporary higher privileges to perform malicious activity. Hence there is a need to detect such attacks during runtime by detecting the user privilege escalation.

Note initially when the user first logged in as a normal user, the ACM has details of the user and his/her privileges in its user_store. Let us now consider how the runtime privilege escalation by the user, by altering the bits in the memory, is detected by the ACM module in the secure VMM. Recall the VMM is used to access and monitor the runtime state information of guest virtual machine such as vCPU registers, process and applications running in the guest virtual machine. There are three different types of memory in the VMM which are known as machine, physical and virtual. Machine memory is the real memory which is controlled by the VMM. Physical memory is the memory assigned to the virtual machine and the virtual machine is under the illusion that the physical memory is the actual memory. The virtual memory is similar to the usage in traditional operating systems. The conversion between machine address to physical addresses is performed using a lookup table in the VMM. The ACM module makes use of the xc_map_foreign_range function in the VMM to access the memory contents of the guest virtual machine. Now the runtime privileges of the logged users are determined by analysing the memory allocated to the virtual machine and the actual privileges of the users are available in user_store. Hence in this case, the ACM module can detect the privilege escalation of the logged users.

Another example attack scenario is when a virtual machine is infected with malware such as conficker [17], torpig [18] or LOIC [19]. In such cases, the IDE module comes into play in the detection of such attacks. This happens when the interactions in the virtual machines are found to be suspicious. For example, the LOIC attack floods the victim machines with TCP, UDP and HTTP messages. Such flooding attacks are detected by the signature or anomaly detection component in the IDE module. For example, such attacks are detected when the traffic from the VM matches with the known attack signatures or exceeds a predefined threshold. The virtual machine is then suspended and taint analysis is performed in an isolated environment.

Secure operation also should consider techniques for secure update of the virtual machines. One approach is to apply the updates to the snapshot image and then validate the image in an isolated environment before applying the updates to the virtual machine in the production environment.

3.3 Secure Migration

Let us now consider the situation when a virtual machine which is running on a one VMM based physical server has to be migrated to another one. It is role of the DEE to ensure that the virtual machine is migrated to a secure platform. If the remote physical server does not have the capabilities to enforce the current virtual machine specific policies, then the virtual machine should not be migrated to the remote location. For example, the DEE needs to ensure that the remote server which needs to host the migrated VM does not have any conflicting VMs already running on it. Similarly, in some cases the remote server may not have hardware support for virtualisation. In such cases, the capabilities of the VMM based security modules may not be capable of enforcing the security policies which require hardware support for virtualisation.

When there is virtual machine migration, the DEE determines whether the set of security policies from the ACM and IDE that are specific to the virtual machine are satisfied. Then the information sharing component in the DEE contacts the remote server to check whether it has the required resources to host the virtual machine to be migrated. Then the capabilities of remote server are checked to ensure that the current level of VM security policies can be enforced at the remote server. Our architecture makes use of TPM based validation to ensure that the remote server is capable of achieving a similar level of security for the virtual machine. An additional challenge arises in the migration of virtual machines between physical servers, when different representations for specification of security policies have been used by the different servers. We have assumed that the specification of security policies have been done using the same language. In our case, we use XML based specification of the security policies for virtual machines.

4 Implementation

In this section we consider a malicious user, with limited privileges, exploiting vulnerability in a virtual machine by performing privilege escalation and then compromising the anti-virus software running in the virtual machine. Then the attacker uses compromised system to generate further attacks. Let us consider how our security architecture is able to deal with such an attack scenario.

In our attack scenario, we have used the anti-virus software Avira [20]. Note that Avira is one of the major anti-virus software providers and is an excellent security product. We have just used this as an example to illustrate how a malicious user to exploit current security measures to conduct attacks. Our research confirms similar attacks are also possible with other anti-virus software.

4.1 Anti-Virus Software Overview

Now let us consider a simplified description of Avira Antivirus Free Software and its security features that is relevant for our illustration purposes. It consists of several Windows services, user-level processes, and kernel-mode drivers. Among these modules, Realtime Protection service is crucial in Avira's protection mechanism, as it provides realtime protection not only to the system (e.g. on-access detection of malware), but also to itself (self-protection such as prevention of unauthorised alteration on Avira-related files). In particular, unloading the kernel-mode drivers and the filter driver is blocked by Realtime Protection service as shown in Figure 2. Also, a user cannot stop, pause, or restart the service, as the service ignores such requests. Also one cannot terminate or kill the processes associated with the service (blocked by the driver).

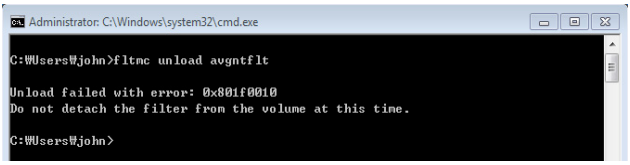

Fig. 2. Driver Protection in Antivirus Tool

Kernel-mode drivers further strengthen Avira's self-protection. One of them locks Avira-related registry keys so that they cannot be modified by the system users. Avira's program folder is protected by the filter driver so that even the user with administrator privilege cannot add any file and delete or modify its files. Furthermore, the processes of Avira are protected by the kernel-driver in a way that even the user cannot kill them. Last but not least, Avira uses files in FAILSAFE folder if some its files in its installation folder are corrupted. To sum up, the protection architecture of Avira has security enforcements to defend it from the malicious users.

Avira also checks for updates regularly and downloads the latest signatures and engines to deal with new types of attacks. There are three ways for carrying out the updates. First, the update of the definitions occurs automatically on a daily basis. Next, a user can trigger an automatic update via a menu or a command-line. Thirdly, a user can download the latest definitions from the Internet, and manually update Avira antivirus with the downloaded definitions. Once an automatic update is started, Avira first checks the current definition and engine versions to determine whether an update is indeed required or not. If so, it downloads the latest definition and engine files from a few dedicated servers, and then checks and installs them. While updating, Avira keeps logging all relevant events so that a user or an administrator can infer possible reasons if the update fails.

4.2 Attack Scenario

In this section, we describe an attack on the file replacement to compromise the anti-virus software and the operating system. Once this can be achieved, the attacker has

complete control of the virtual machine and hence can use the system to generate different types of attacks. We will present an attack scenario where the attacker uses a compromised virtual machine to flood the network with malicious traffic and describe how our architecture can prevent such attacks. Our experiment environment was a virtual machine with Windows 7 Ultimate with Avira Free Antivirus 2012 which was running on Xen VMM

We have used a staged malware in this scenario: First the malicious user runs a malware installer (first stage); the malware installer performs only the actions that are permitted under any anti-virus software's realtime protection. In this particular example, it checks the following: OS version, privilege of current user, anti-virus solution installed on the system, and version of currently active signatures and engines. If Avira is installed on the target system, then the installer triggers an update; alternatively, s/he may just wait for an update to be started by the Avira's scheduler service.

When an update begins, the installer monitors the status of Avira's Realtime Protection service. Once the service is deactivated during the update, the installer performs the required actions that are normally blocked or prevented by Avira's Realtime Protection service. In this example, the installer's ultimate goal is to replace Avira's sqlite3.dll with a malicious one (second stage) so as to subvert both Avira and the system.

Fig. 3. Real-time Protection Service Process after update

It performed the following tasks:

- For privilege escalation, it dropped and executed any known or zero-day exploit that is normally detected by Avira. Notice that this local privilege escalation (e.g. from admin to SYSTEM) is required only once. After this file replacement process, the malware obtains SYSTEM privilege on the target machine.
- Unloads Avira's filter driver that is normally protected by the service.
- Dropped the real payload (fabricated sqlite3.dll) and replaced the original file in Avira's installation folder with the malicious one. This file operation is shown in Figure 3.
- The installer deletes itself as a clean-up process to erase its existence; alternatively, the payload may delete the installer.

As the filter driver has been unloaded, it should be restored, even though Realtime Protection service automatically loads and attaches the filter driver when it restarts. The reason for the restoration is that the service's restart triggered by Avira after an update proceeds to some extent and fails if the driver remains unloaded; of course, the installer can manually restart the service after the first restart by Avira fails. But still

the best solution is to restore the driver, because the service restart by Avira succeeds if the filter driver is restored. Interestingly enough, even if the start of the service was triggered and failed, it is logged as successfully started in the Avira's update log file, which is good from the attacker's point of view.

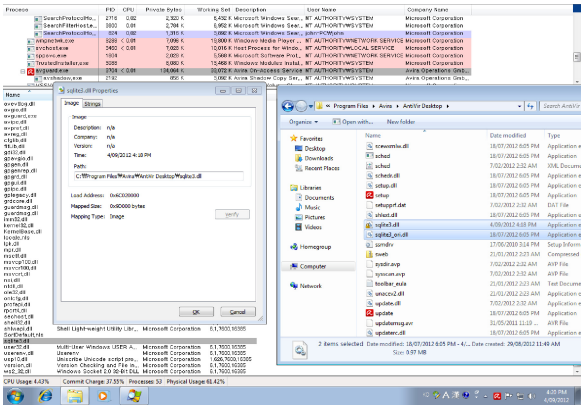

Fig. 4. Successful Attack

On restarting, Realtime Protection service loads the malicious sqlite3.dll which provides full SQLite functionalities, and becomes active without any problem. However, once loaded by the service, the malicious sqlite3.dll obtains SYSTEM privilege on the target machine. In other words, this attack allows the malware to escalate its privilege from a user to SYSTEM, which means UAC (User Access Control) on Windows becomes ineffective. Also, almost any malicious activity becomes possible, as it is loaded and executed in the context of Avira's Realtime Protection service. Furthermore, the DLL can perform file operations on the installation folder, even while the filter driver is loaded; this allows the attacker to update the malicious DLL. The result of this file replacement and loading operations is shown in Figure 4. The original sqlite3.dll (sqlite3_ori.dll, 389KB) has been replaced with the malicious version (sqlite3.dll, 612KB), and the fabricated DLL has been loaded by the service (window on the left side). Here, the original DLL was not removed to show its replacement. After becoming a part of Avira, the malware might be able to modify Avira's memory area. If so, it is possible to make Avira to look normal (with the tray icon's umbrella open), but totally ineffective.

In the above scenario, the TPM prevents such unauthorised services during restart of the service. However it is important to note that the attacker can also generate attacks by compromising his virtual machine during runtime and generate attacks without restating the service. In our architecture, such runtime attacks are prevented by the secure VMM when the ACM module detects the privilege escalation of the logged user to system level or when the IDE module detects malicious traffic originating from the compromised virtual machine. Although the attacker is successful in compromising the virtual machine, s/he does not have access to the security components in VMM. Hence such attacks will not be successful with our integrated security architecture.

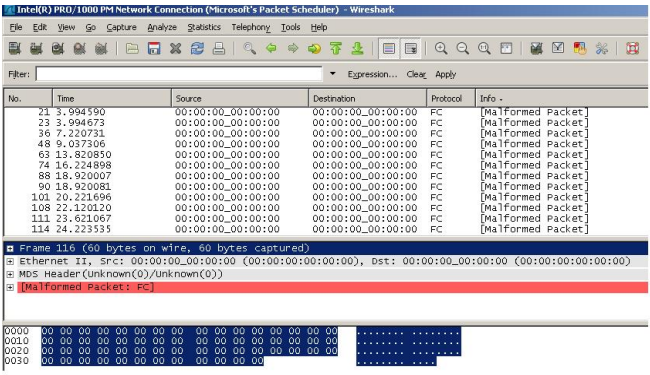

Fig. 5. Flooding with Malicious Traffic

Now let us consider an attack scenario. Figure 5 shows the case where an attacker has compromised a virtual machine during runtime and generates malicious traffic without restarting the virtual machine. Such an attack will be successful and the attacking source can remain anonymous in a traditional datacenter. Since the attacker has obtained complete control of the virtual machine and the traditional security tools in the virtual machine, s/he can alter the logs in the compromised system. Hence it is extremely difficult for the datacenter administrator to determine the attacking source for such attacks since the attack traffic does not have any valid MAC or IP address.

With our architecture, the attacks shown in Figure 5 are not possible in the first place. Since the traffic does not have valid MAC or IP address it will be blocked by the IDE module and an alert will be raised to the administrator. Hence our architecture can detect and prevent such an attack even before the attack traffic is placed on the network.

5 Concluding Remarks

We have proposed integrated security architecture which combines trusted computing, access control and intrusion detection techniques for securing the life cycle of the virtual machines. We have also shown how our architecture can prevent attacks from malicious users exploiting vulnerabilities to achieve privilege escalation and then using the compromised machines to generate further attacks.

References

1. Ferraiolo, D., Kuhn, R.: Role-based access control. In: Proceedings of the 15th NIST-NCSC National Computer Security Conference, pp. 554–563 (1992)
2. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A logical language for expressing authorizations. In: Proceedings of the 1997 IEEE Symposium on Security and Privacy. IEEE Computer Society, Washington, DC (1997)

3. DeTreville, J.: Binder: A logic-based security language. In: SP 2002: Proceedings of the 2002 IEEE Symposium on Security and Privacy, p. 105. IEEE Computer Society, Washington, DC (2002)
4. Li, N., Mitchell, J.C.: Rt: A role-based trust-management framework. In: Proceedings of the Third DARPA Information Survivability Conference and Exposition, pp. 201–212. IEEE Computer Society (2003)
5. Herzberg, A., Mass, Y., Michaeli, J., Ravid, Y., Naor, D.: Access control meets public key infrastructure, or: Assigning roles to strangers. In: SP 2000: Proceedings of the IEEE Symposium on Security and Privacy. IEEE Computer Society, Washington, DC (2000)
6. The Open Source Network Intrusion Detection System: Snort, http://www.snort.org/docs/iss-placement.pdf
7. Smith, J.E., Nair, R.: The Architecture of Virtual Machines. IEEE Internet Computing (May 2005)
8. Brewer, D.F.C., Nash, M.J.: The Chinese Wall security policy. In: Proc. of IEEE Sympoisum on Security and Privacy, pp. 206–214 (1989)
9. Sailer, R., Jaeger, T., Valdez, E., Caceres, R., Perez, R., Berger, S., Griffin, J.L., van Doorn, L.: Building a MAC-based security architecture for the Xen open-source hypervisor. In: Proceedings of the 21st IEEE Annual Computer Security Applications Conference, Washington, DC, USA (2005)
10. Bell, D.E., La Padula, L.J.: Secure Computer Systems: Unified Exposition and Multics Interpretation. ESD-TR-75-306, MTR 2997 Rev. 1, The MITRE Corporation (March 1976)
11. Dunlap, G.W., King, S.T., Cinar, S., Basrai, M.A., Chen, P.M.: ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. In: Proceedings of OSDI (2002)
12. Garfinkel, T., Rosenblum, M.: A virtual machine introspection based architecture for intrusion detection. In: Proceedings of NDSS (February 2003)
13. Jones, S.T., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: VMM-based hidden process detection and identification using Lycosid. In: Proc. of ACM VEE (March 2008)
14. Trusted Computing Group, TCG Specification, Architecture Overview, Specification Revision 1.2 (April 2004), http://www.trustedcomputinggroup.org
15. Newsome, J., Song, D.: Dynamic taint analysis: Automatic detection and generation of software exploit attacks. In: Proceedings of NDSS (February 2005)
16. Litchfield, D.: Threat Profiling Microsoft SQL Server, http://www.cgisecurity.com/lib/tp-SQL2000.pdf (last viewed: July 31, 2013)
17. Seungwon, S., Guofei, G.: Conficker and Beyond: A Large-Scale Empirical Study. In: Proceedings of the 26th Annual Computer Security Applications Conference, Austin, Texas, USA, December 6-10, pp. 151–160. ACM Press, New York (2010)
18. Stone-Gross, B., Cova, M., Gilbert, B., Kemmerer, R., Kruegel, C., Vigna, G.: Analysis of a Botnet Takeover. In: Proc. of IEEE Symposium on Security & Privacy, vol. 9(1), pp. 64–72 (2011)
19. LOIC, http://sourceforge.net/projects/loic/
20. Avira Antivirus Software for home and business, http://www.avira.com