Securing Services in Networked Cloud Infrastructures

Vijay Varadharajan and Udaya Tupakula

Abstract— In this paper, we propose techniques and architecture for securing services that are hosted in a multi-tenant networked cloud infrastructures. Our architecture is based on trusted virtual domains and takes into account both security policies of the tenant domains as well as specific security policies of the virtual machines in the tenant domains. We describe techniques for detecting a range of attacks such as attacks between the virtual machines within a trusted virtual domain, attacks between the virtual machines in different domains, malicious insider attacks and attacks against specific services such as DNS, database and web servers within a domain. We address security policies for trusted virtual domain management such as secure addition and deletion of a virtual machine and the revocation of privileges associated with a virtual machine in a domain. We also discuss forensic analysis of attacks and fine granular detection of malicious entities and mechanisms for restoration of services. Furthermore the proposed architecture provides mechanisms for enhancing the assurance of communications between the virtual machines in different domains. Finally, we present the implementation of our security architecture using Xen and illustrate how our architecture is able to secure services in networked cloud infrastructures.

Index Terms—Networked Cloud Security, Security Architecture, Security Attacks, Trusted Virtual Domains, Security Management

1 INTRODUCTION

TODAY most information services are hosted online and accessing online services has become part of everyone's daily life. For example, users could be using them to read today's news with rich content or doing online banking with complex functionalities. On the other hand, online services have become a constant target for malicious attackers trying to exploit vulnerabilities to obtain sensitive data to benefit them financially or to gain access to systems which can then be used to crate further attacks on other systems. Hence it has become a nightmare for the service operators to ensure the security, quality and availability of their services.

Cloud computing [1] offers several advantages to organisations to suit their dynamic requirements for hosting online services in the Internet. For example, cloud computing promotes on-demand self-service, broad network access, rapid elasticity, pay for use and high availability. Hence different sectors including healthcare, utility, social networks and government are migrating their services to the cloud environment. There are different types of cloud models such as public cloud, private cloud and hybrid cloud, as well as different types of deployment infrastructures such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1]. In this paper, we will be only concerned with the IaaS cloud infrastructure.

Although cloud offers several advantages, it also poses certain significant challenges. Clearly security and privacy challenges arise when there are multiple tenants with different policies and requirements using the cloud infrastructure. When sensitive data belonging to enterprises and individuals are stored and used by services in the cloud, it poses security as well as privacy issues. There are also significant security issues arising out of malware and attacks in the cloud which not only have access to both data and services of many users but also the ability to propagate to many systems over the cloud infrastructure. Then there is the issue of trust on the cloud providers themselves. In addition to such security, privacy and trust issues, large scale cloud infrastructures can be vulnerable to environment disasters such as earthquakes, flooding and blackouts. Hence not only for security but also for disaster recovery reasons, reputed organizations as part of their business strategies are hosting their services in multiple cloud infrastructures which are in separate geographical locations to achieve higher availability and dependability of services. This in turn leads to increased complexity when it comes to securely managing their infrastructure and services hosted in multiple clouds environment. This is where the concept of trusted virtual domains (TVDs) [2, 3] has been introduced to simplify the management of services that are hosted in multiple clouds in separate geographical locations.

A TVD enables grouping of related virtual machines running on separate physical machines into a single network domain, which trust each other based on a common security policy. The policy is uniformly enforced, independent of the physical boundaries. TVDs use virtualization and overlay technologies to provide confinement boundaries for a protection execution environment that are typically distributed over several physical platforms. While conceptually simple, the realisation of TVDs pose significant security challenges such as integration of sophisticated security techniques such as secure network virtualization, trusted channels, distributed security management and secure hypervisors. For instance, there can be different types of security threats where a virtual machine can be a member of multiple domains. The attacker can exploit a single vulnerability in any of the operating system or applications in a virtual machine to generate attacks in all the virtual domains in which the vulnerable virtual machine is a member of.

In this paper, we propose techniques and develop architecture for securing the TVDs in the context of multiple cloud infrastrcutures. Our architecture makes use of different technologies

• *Vijay Varadharajan is with the Computing Department, Macquarie University, Australia. E-mail: vijay.varadharajan@ mq.edu.au*
• *Udaya Tupakula is with the Computing Department Macquarie University, Australia. E-mail: udaya.tupakula@mq.edu.au*

such as trusted computing [4], hypervisors [5], virtual introspection [6], dynamic and static analysis based attack detection for securing the TVDs. We need different techniques to detect attacks in the current environment where the emerging attacks exhibit multifaceted malicious behaviours. Using a single form of detection/prevention technique is not sufficient to deal with such emerging attacks. For example, Stuxnet was designed to be spread through malicious insiders using USB, obtain access to critical components using default passwords, and then exploit 4 zero day vulnerabilities in Windows systems. Such attacks are looking for a range of weak points in the current systems, which are more likely to arise in a multiple cloud environment. However, on the other side of the coin, if a malware exhibits a range of malicious behaviours, then potentially it can become easier to detect the attacks.

The main contributions of this paper are: i) Detailed analysis of security attacks in virtual domains, leading to a comprehensive attacker model for trusted virtual domains in a multiple cloud environment. We consider a range of attacks such as attacks between the virtual machines within a TVD, attacks between the virtual machines in different TVDs, malicious insider attacks, external attacks, and attacks on specific services such as DNS, database and web servers within a TVD; ii) A modular security architecture for securing trusted virtual domains: We develop modular security components for enforcing different security policies and techniques such as secure communications between virtual machines, whitelisting of applications and processes for virtual machines, validating the state of the virtual machines, validating input and outputs of critical applications and services, and detecting attacks using signature based, anomaly based and domain based policies; iii) Techniques for detecting zero day attacks: We show how the security components of our architecture can be used to detect and/or prevent zero day attacks. We have used MAEC [7] framework to characterise these attacks; iv) Techniques for achieving assurance for the communications in a virtual domain based multi cloud environment. We demonstrate how our architecture can be used to provide assurance against scenarios such as attacker conducting malicious communications using a compromised virtual machine, attacker erasing the legitimate communication details from the virtual machine to hide his/her malicious behaviour, and malicious clients claiming unauthorised communications from the TVD virtual machines.

The paper is organized as follows. In Section 2, first we present an overview of the TVD and discuss the attacker model and the security objectives of our TVD architecture for multiple clouds. Then we present a high level overview of the proposed security architecture and how it helps to meet the overall security objectives. Section 3 presents a detailed discussion of the security architecture and the security components that are deployed on each physical platform for securing the TVD virtual machines in multiple clouds. Section 4 describes the implementation of our security architecture using open source Xen based virtual machines and discusses how the proposed architecture deals with different types of attacks. Section 5 presents some of the relevant related works and compares them with our work. Finally, Section 6 concludes the paper.

2 Our Approach

2.1 Assumptions and Attacker Model

Consider a scenario where an organization uses multiple IaaS clouds in the provision of its services to its customers. In such a scenario, there are multiple cloud service providers, each of which has its own cloud system administrators. The tenants host their services in the virtual machines that run on top of the Virtual Machine Monitor (VMM) or hypervisor [5]. A VMM is an additional software layer that enables to run multiple operating systems on a single scalable physical computer. There are tenant administrators who manage the tenant virtual machines. In Figure 1, the tenant virtual machines are hosted in multiple cloud environments. A TVD enables grouping of related virtual machines running on separate physical machines into a single network domain. We have a TVD Administrator (tenant administrator) who is responsible for defining the security policies for a given TVD, which are enforced on all the virtual machines belonging to the TVD owned by a tenant. Then there are tenant users (or tenant's customers) who use the applications and services running in the tenant virtual machines. Examples of cloud providers include Microsoft Azure [8] and Amazon Web Services (AWS) [9]. The cloud system administrators are individuals from these corporations entrusted with system tasks and maintaining cloud infrastructures, who will have access to privileged domains. We assume that as cloud providers have a vested interest in protecting their reputations and resources, the adversaries from the cloud provider perspective are malicious cloud system administrators[1]. For example, a tenant (a company) may be developing a service using virtual machines in Microsoft Azure cloud [8]. The company may also be hosting some other applications using virtual machines in the AWS cloud [9]. The company may also be doing all its finances in its own private cloud. The company (tenant) needs to manage all its virtual machines and services and applications in all these different clouds.

One of the common assumptions often made in a TVD based environment is that free communication is permitted between the virtual machines *within* a trusted virtual domain. However there are threats within a domain and if intra communications within the domain are not monitored for attacks, then an attacker can potentially attack all the virtual machines within the TVD spanning multiple physical platforms in different cloud infrastructures. Furthermore, since a virtual machine can be a member of multiple TVDs, the attacks are possible in all the TVDs where the compromised VM is a member of. Hence the attack surface can be considerably large with the TVDs. Hence we believe that there is a need to enforce fine granular security both *within and in between* TVDs.

The attacker model considers who can access communication channels and compromise components of the TVDs.

(i) First, we assume that the VMM is secure and trusted and is part of the trusted computing base (TCB). We ensure that at boot time the VMM is loaded in a secure state using Trusted Platform Module (TPM) based attestation. At runtime, we assume that that TCB on each platform cannot be compromised but can be replaced or altered between two bootstrapping phases (e.g. binaries modified) which needs to be detected by

[1] We envisage the existence of malicious cloud system administrators; this is despite the cloud providers having processes and methodologies to ensure the trustworthiness and integrity of their cloud system administrators.

our architecture.

(ii) Second, the virtual machines belonging to a domain (tenant) can be compromised by the downloading and installation of unauthorized applications or malware. We will collectively refer to different types of attacks such as buffer overflows, injection, redirection, worms, viruses, Trojans and rootkits as malware attacks. The compromise of virtual machines can lead to them being used as bots to carry out further attacks. The attacks themselves could be zero day attacks, for which no previously known solutions or patches exist at the time of attacks.

(iii) There can also be attacks from tenant users (customers). Consider, for instance, a tenant which is a software development company making use of cloud resources. Although the TVD (tenant) administrators could have provided host based security tools in their tenant virtual machines, a malicious tenant user (tenant employee) may be able to circumvent such security tools. Also there could be a need for higher privileges to certain users within an organisation. For example, software developers may legitimately require higher level administrative privilege as part of software development process. Hence we need to authenticate users and have access control mechanisms to assign different privileges.

Malicious users and processes can misuse these higher privileges to generate attacks. Furthermore, it is common for users to use different devices such as USB devices for sharing files. Hence malicious insiders can include malicious files in such shared devices to compromise multiple systems within an organisation (eg. Stuxnet). It could also be that some internal entities (insiders) are deliberately malicious whereas others may not be security conscious and resort to installing applications without the knowledge of security procedures. Furthermore, use of certain applications such as Instant Messaging, Skype and Facebook make the user machines more vulnerable to attacks since malicious files can be easily embedded within these communications and distributed to users within an organisation even without their knowledge.

The cloud provider needs to provide secure isolation between the tenant virtual machines. However the cloud service provider may not be aware of the operating systems and applications running in a tenant virtual machine. Hence it is not an easy task for the cloud service provider to enforce security policies on the tenant virtual machines. Furthermore since the elastic nature of cloud allows the ability to dynamically increase the resources allocated to tenant virtual machines, the attacker can use this capability in compromised tenant virtual machines to generate sophisticated attacks.

(iv) There could be attacks such as eavesdropping or manipulation on communication channels thereby allowing the attacker to gain or modify information in an unauthorised manner. Hence there is a need for secure channels both within members of a TVD and members in different TVDs. This is particularly important when policy changes need to be communicated, e.g. when a user and a tenant virtual machine belonging to a user is revoked of certain privileges. There is a need to secure the policy management procedures and the communication of policy update mechanisms to prevent an adversary attacking the policy update mechanisms. This could be done for instance the attacker impersonating the trusted TVD component which is responsible for updating and distributing the security policies.

2.2 Design Goals

The main security objectives of our TVD architecture for multiple clouds are as follows:

- Attack Detection: Techniques for detection of a range of attacks such as attacks in the TVD architecture. These include attacks between the virtual machines within a TVD, attacks between the virtual machines in different TVDs, malicious insider attacks, external attacks, and attacks against specific services such as DNS, database and web servers within a TVD.
- Secure TVD Management: Management of virtual machines within a trusted virtual domain should satisfy well defined security policies. For instance, the addition and deletion of a virtual machine and the revocation of privileges associated with a virtual machine in a TVD should be subjected to specified security policies.
- Secure TVD Communications: Techniques for securing communications between VMs in different TVDs and mechanisms for enhancing the assurance of communications between the VMs in TVDs.

2.3 Architecture Outline

In our architecture, the TVD administrator is responsible for defining the security policies for a given TVD. These polices are enforced on all the virtual machines belonging to the TVD. We assume that the TVD administrator makes use of the TVD Master for coordinating and enforcing its domain policies like in the previous approaches [10-12]. The TVD Master has therefore security policies that determine the membership and revocation of virtual machines and platforms in the grouping. For instance, a TVD Master may have security policies that prevent a virtual machine to be hosted on a platform in a cloud that already has virtual machines belonging to another competitive organization. We assume that TVD Master is trusted and secure and that there is co-operation between the TVD Master and the cloud service providers.

Note that a virtual machine can be a member of multiple TVDs. For instance, consider a scenario where an auditor is a consultant to two customers and hence the virtual machine belonging to that auditor can be the member of two TVDs of the two tenants. In this case, the communications of the auditor's system must meet the specific requirements of each TVD of which it is a member of.

The TVD administrators can use existing security tools such as host based security tools for securing their virtual machines. Since the tools are placed in the monitored host, though they have excellent visibility into the state of monitored virtual machine, and hence can detect attacks more efficiently (compared to other network based tools), these tools themselves are vulnerable to attacks [13]. The security attack monitoring tools themselves need to reside in a trusted and secure environment within the VMM having good visibility into the state of virtual machines while remaining protected from attacks.

The TVD Master determines the domain security policies and makes use of Smart Security Module (SSM), one in each physical platform hosting a TVD, to co-ordinate and enforce the TVD security policies on its virtual machines. If virtual machines belonging to different tenant TVDs are hosted on the same physical platform, then one SSM is implemented for each TVD. In Figure 1, VMM1 has two SSMs since it is hosting virtual machines that

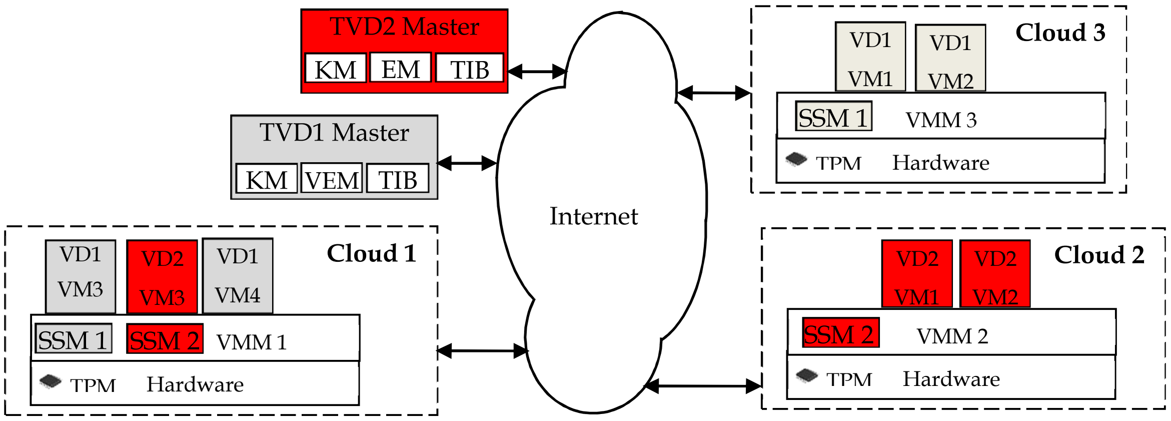


Fig. 1. Trusted Virtual Domain Scenario

belong to two different TVDs. VMM2 and VMM3 have a single SSM since they are hosting virtual machines that belong to a single TVD.

Whenever a virtual machine needs to be instantiated within a TVD in a VMM, the SSM checks whether the TVD policy is satisfied before allowing the VM to join the TVD and then monitors the VM against different types of attacks. The admission of a physical platform is controlled by the TVD Master.

The SSM forms part of the TCB and resides within the VMM in the physical platform. Each physical platform with a VMM that is able to host a TVD is equipped with a TPM [4] (and hence referred to as a trusted platform). TPM is responsible for determining and storing the integrity measures (e.g. during the bootstrapping procedure).

Now we briefly mention the different components of the SSM (shown in Fig 2). SSM consists of the following components: Integrity Management (IM), State Analysis Report (SAR), Entity Detection (ED), Store and Restore (SR), Entity Validation (EV), Virtual Domains (VD), Local Key Management (LKM), and Report Generation (RG).

The purpose of IM is to ensure that the virtual machines and its applications are secure during boot time. IM helps to detect unauthorized changes to existing applications and prevents them from running in the virtual machine. SAR is used to generate the VM status report and ED is used for identifying the entities that are generating or receiving the traffic in the virtual machine. The SR component captures the VM specific knowledge such as the operating system, applications, updates and resources allocated to the virtual machine. SR is used to conduct forensic analysis of attack and fine granular detection of malicious entity. SR enables auditing of the VM communications and quick restoration of the virtual machine service in case of attacks. It also flags all the new processes/applications for validation by the TVD Master. EV is used to enforce VM specific security policies using signature and anomaly based detection and input/output validation. VD is used in the enforcement of domain specific policies. LKM is used for storing the keys that are required for secure communications of virtual machines. Secure Data Exchange (SDE) [14] is used for securing the communication of TVD virtual machines that are hosted in different servers and clouds. Since one cannot guarantee attack free operation in such a cloud environment, VM communications create a report which can be analysed for malicious behavior. RG is used for generating reports on VM communications. This report is transferred to the TVD Master which performs the analysis and uses the analysis to provide assurance of the VM communications. Such a mechanism enables a victim tenant to report a malicious communication to the TVD Master. Although the communication report is generated by the local SSM, the TVD Master provides the assurance by issuing the certificate for each SSM. The TVD Master decides which components are instantiated at each SSM.

Before going through the architecture in detail in the next section, we now briefly sketch how the components outlined above can help to achieve the security objectives mentioned in Section 2.2.

Attack Detection: SSM detects the attacks by considering the specific features of the virtual machine and domain policies. The VM specific policies are developed by considering the operating system, applications running in the virtual machine and the resources allocated to the virtual machine. The SSM proactively prevents malicious communications by validating the runtime state of the virtual machine and the traffic originating from the virtual machines. The SSM components monitor the resources used by each virtual machine such as memory, CPU, network, disk space and all the interactions between the virtual machines. If any of the SSM components observes suspicious behavior of any virtual machine, then that virtual machine is isolated from the TVD and an alert is generated to the TVD Master. However since the proactive prevention does not provide guarantee of attack free communications, it also provides a reactive approach to deal with the attacks. The RG in the SSM generates a report for each VM communication. The victims use this report for providing a feedback on the malicious VM communication.

Secure TVD Communications: The TVD Master specifies the domain wide policies such as key management (KM) for securing the intra and inter domain communications, virtual entity management (VEM) policies, and policies for maintaining the domain wide trusted information base (TIB). KM is used for storing and establishing keys for secure communications both within a TVD and between TVDs. We assume that each TVD Master has a public key – private key pair. We also assume that each SSM has a public key private key pair. TVD Master and SSMs interact to establish a symmetric session key to protect virtual machine communications. The component VEM contains information on the virtual machines within a domain. This information includes the number of virtual machines in the domain and their location, hosted services in the virtual machines, users, their roles and the specific services that can be accessed by them. This information is used in the specification of policies for attack detection in each of the virtual machines. We make use of IDMEF [15] for the specification of policies. For example, a web server and a DNS server in a TVD can be made to have public access whereas payroll server can only be accessed by employees. The TIB component contains information about other TVD Masters and their policies for sharing of information between domains. For example, services that can be accessed in the different TVDs, policies that need to be enforced for inbound/outbound communications for different TVDs and assurance reports that need to be generated for communication between different domains. TIB also stores security attributes for each association within the TVD. The attributes include security session key and security services needed for that association. The TIB is implemented as table of entries, one for each communicating pair of hosts. This information determines the security policies that need to be enforced by the SSM. Our architecture makes use of the IEEE Standard for Interoperable LAN/MAN Security (SILS) [14] for securing the communications of virtual machines within a TVD. However SDE does not prevent malicious communication resulting due to the compromise of virtual machines within the domain. Hence we have developed additional techniques for monitoring the virtual machines and securing their communications. These are described in Section 3.

Secure TVD Management: The TVD Master and SSM manage the addition and deletion of a virtual machine to/from a TVD. In the case of a VM addition, first the TVD Master determines the policies that need to be satisfied in terms of the specific cloud infrastructure and platform to which the VM needs to be added to. If the cloud service provider is able to host the new VM on a platform that is already running a VM belonging to the TVD, then the VM addition is straightforward as there is already a SSM for that TVD in that platform. If the cloud service provider is unable to find resources on the platforms that are already running one of the VMs belonging to that TVD, then a new SSM has to be created on a platform before adding the new VM. In the case of a VM deletion from a TVD, if there are multiple VMs belonging to the TVD running on the platform, then the SSM in that platform just terminates the VM. If there is only a single VM belonging to the TVD running on the platform, then the SSM terminates the VM and the TVD Master then terminates the SSM running on that platform.

Enhanced Assurance Mechanisms: The TVD Master is able to provide assurance of TVD's services to its tenants. In general, the assurance from a cloud service provider is more useful for small tenants. In the case of large tenants, often self-assurance is more useful due to their reputation. Even in the case of large tenants, the assurance from the TVD Master has a role to play. For example, if the customers of a large reputable tenant express their concerns regarding the use of cloud services, then an assurance from the TVD Masters guaranteeing certain properties of the cloud services rendered can be valuable for the customers, especially since the communications can span multiple cloud infrastructures. For example, if a server VM in TVD1 provided some service to the client VM in TVD2, then the TVD1 Master and TVD2 Master will generate the communication reports.

3 Smart Security Module

In this section, we consider the design and functionalities of the various components of the SSM in detail. SSM is used for local management of TVD virtual machines hosted on a trusted platform. Figure 2 shows the logical architecture of the SSM. SSM design consists of modular security components and supports parallel processing for better performance. The TVD Master determines the security policies enforced at each SSM.

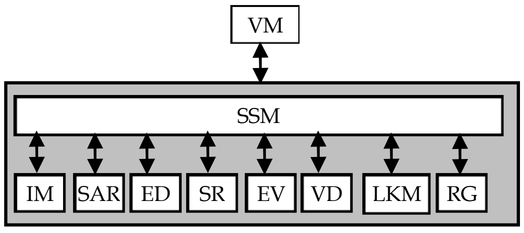


Fig. 2. SSM Components

3.1 Integrity Management

The IM is used for integrity management [16-18] of the virtual machines hosted on each platform. It ensures that the virtual machines boots into a known trusted state. After a machine is rebooted, a snapshot of the machine is taken before performing any interaction. This helps to minimize the restore time in case of any attack on the virtual machine. The IM supports property based attestation of the TVD virtual machine using TPM [17, 18]. A component or a platform is considered to be trusted if it satisfies the necessary hash measurements as well as the property specifications. A property specification may make direct use of a certificate issued for that property or by validating other properties required to satisfy the given property.

In this vein, we have developed a simple logic language (called ALOPA) [17] required to express property relationships. Using ALOPA, it is possible to determine whether a component or a collection of components or a platform or satisfies a given property or sets of properties using ALOPA rules. A full description of the ALOPA language and its rules can be found in [17].

The next step is to identify the properties for the services offered by the cloud service provider. We have made use of the assessments from the Cloud Security Alliance (CSA) [19] to derive these properties. The CSA has designed a self-assessment framework for cloud providers to publish their cloud platform's security controls and capabilities. The self-assessments from different cloud service providers can be found in [20]. We use these CSA self-assessment capabilities to derive property specifications in our architecture.

Table 1 shows some properties that are used in our architecture. Different parties can make claims with respect to these properties. For example, cloud service provider can claim the property P1 by permitting 3rd party audits. In fact, the same property could be claimed by the tenants as well if the tenant allows its virtual machines to be audited by a 3rd party. In our architecture, the TVD Master can use the first property to state that for all the VMs in its domain, audits are available, which are either self-certified by the cloud service provider (CSP) or by a 3rd party. That is, the policy of the TVD Master allows the use of self-certification by the cloud service provider or 3rd party's audit certifications. The second property P2 is concerned with the use of encryption to secure communications between the TVD

VM and a customer. The third property P3 ensures that the tenant virtual machine has anti-virus software installed. This can be used to reason that certain types of malware can be detected and prevented in the virtual machines. Note that the tenants can use 3rd parties as validation authorities. For example, the tenant can be certified of the anti-virus property from a 3rd party anti-virus provider on the version of the software and its last update. In this case we assume that the anti-virus vendor acts as CA for its customers. The customer's machine (say VM1) validates the integrity of the anti-virus program before executing it and the measurements are stored in the PCR. Consider that a new VM2 makes Attestation request and VM1 reports all the measurements to VM2. Now VM2 can forward the measurements to the CA to validate if VM1 is in secure state. Now the CA can validate these measurements and confirm if the anti-virus is running the latest version of the tool. The property P4 is concerned with either the cloud service provider or the TVD Master providing assurance guarantees for VM transactions.

TABLE 1

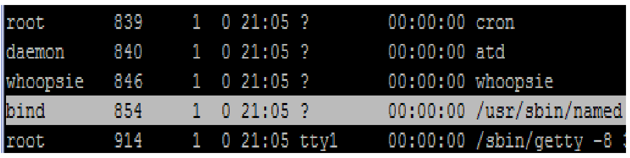
Pr ID	Properties	Validation Authorities
P1	CSP or Third party certified audits available	CSP Self-certified, 3rd party
P2	Encryption supported between VM and Users	Self-certified
P3	Antivirus software installed	Self-certified, 3rd party
P4	Assurance on the TVD-VM communications	CSP, TVD Master

The IM validates the properties of the VMs during boot time. However the virtual machines can be compromised during runtime; hardware failures (including TPM) are frequent in the cloud scale and virtual machines migrate between different physical servers. Hence there is a need for additional monitoring of the VM runtime state and VM interactions with other hosts to detect various attacks. For example, some processes can be altered during runtime or even some critical processes (e.g. belonging to the security tools) can be terminated by the attackers; hence the need for other components of SSM in our architecture.

3.2 State Analysis Report

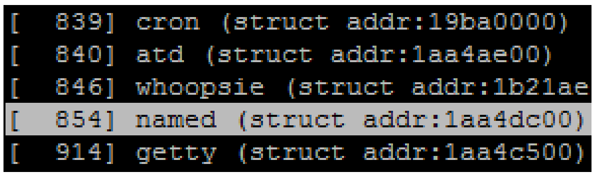
SAR is used for generating reports on the state of the virtual machines. The report is updated whenever there is a variation in the processes running in the virtual machine. The variation trigger for processes is obtained from the CR3 register value which can be used to identify process initiation and termination events. This also helps to identify the execution context of system calls and get the state of the virtual machines.

SAR parses the VM's memory and captures VM runtime information such as processes that are running in the virtual machine, parent processes, path information related to the processes and open ports on the VM. It is similar to the report generated by the task manager in the Windows or top in Linux. The main difference is that the report is generated by a component residing within the VMM instead of being inside the virtual machine. Hence we take this report to be more trusted. Figure 3 shows the trimmed version of processes running in the DNS server where the information is generated using the top tool in the VM; Figure 4 shows similar information related to the processes and the corresponding address location for each process



```
root      839    1  0 21:05 ?       00:00:00 cron
daemon    840    1  0 21:05 ?       00:00:00 atd
whoopsie  846    1  0 21:05 ?       00:00:00 whoopsie
bind      854    1  0 21:05 ?       00:00:00 /usr/sbin/named
root      914    1  0 21:05 tty1    00:00:00 /sbin/getty -8
```

Fig. 3. DNS State report using tools in VM



```
[  839] cron (struct addr:19ba0000)
[  840] atd (struct addr:1aa4ae00)
[  846] whoopsie (struct addr:1b21ae
[  854] named (struct addr:1aa4dc00)
[  914] getty (struct addr:1aa4c500)
```

Fig. 4. SAR report on DNS VM state

that is obtained from the SAR component.

With malware such as Conficker, Torpig and rootkits, the attackers typically disable the security tools and/or install malicious programs such as backdoor and alter the tools (such as task manager, dir, top and ls) so that they do not report the malicious behavior. For example, the attacker can install a backdoor with hidden processes and alter the tools and not report the backdoor program. The users and even the administrators may not be aware of the compromise of their systems. However the attacker will not be able to alter the reports generated by the SAR since they are generated in the trusted VMM. Hence the analysis of the SAR produced report helps to identify the malicious activities performed by malware attacks such as disabling of the host based security tools in the VM or backdoor programs installed in the VMs.

3.3 Entity Detection

The entity detection (ED) is used for detecting entities that are generating the attack traffic. Whenever a new flow is initiated by a virtual machine, the traffic is received by the ED component. We represent each communication as a sequence of packet flows (P) between the specific ports and IP addresses of the source VM and the destination VM, ($S \rightarrow D$: P1, P2…PN). The ED differentiates between different flows based on the protocol header information in the TCP/IP protocol stack. For each flow, the ED determines the information about the entity that initiated the flow, e.g. an application or a process.

We perform a simple lookup with the recent SAR report to determine the entity that is generating or receiving the traffic. For outgoing packets, the entity that is responsible for originating the traffic is determined from the source port information in the TCP/IP header. Similarly for incoming traffic, the entity that is responsible for receiving the traffic is determined from the destination port of the traffic. There is a negligible overhead since it is only necessary to map the traffic to the process that is linked to the specific port.

If any of the process is found to be infected or malicious at later stage, then that specific process is terminated. Hence ED helps to deal with the attacks at a fine granularity level. Note that the SSM modules are optional. Hence the granularity of the entity can vary depending on the invoked modules. For example, the TVD administrator may not want to invoke SAR on a particular server. In this case, the granularity of the entity is virtual machine rather than the process that generated the traffic.

3.4 Store and Restore

The SR component is used for capturing the knowledge of OS and applications running on each VM. This is useful for detecting VM specific attacks, auditing virtual machine communications as well as for restoration of VM services after the attacks.

We use a MySQL database for storing the VM related information. Before a VM joins the TVD, the TVD administrator uploads generic details of the VM into the SR. For instance, in the case of Windows XP, SR has information on Windows XP image, service pack version, Internet Explorer version, any additional applications installed on the VM and resources allocated to the VM. The reports generated by the SAR are used to track the changes to the OS and applications in the virtual machine.

We have obtained VM specific information for some common services such as DNS, Web server and database server. For example, we have analysed the default processes for authoritative DNS server running on a Linux OS. The authoritative name servers are the original sources for all the DNS resolutions for a zone. The hosts can make resolution requests for the services within the domain. We have analysed the default processes for The Berkeley Internet Name Daemon (BIND) 9.8.1-P1 running as the DNS on Linux server with 3.5.0-23 kernel. There are total of 59 processes in the clean state installation of DNS on Linux. Figure 3 shows partial list of the processes running in the DNS server virtual machine and the highlighted process which is located in the specified path (/usr/sbin/named) is responsible for receiving and responding to the client resolution requests. The installation of a host based security tool in the DNS server adds three new processes. The new processes are validated by the TVD administrator. We maintain the details of these 62 processes in the SR and use this as a reference to monitor the DNS server during runtime.

The clean state information obtained from the SAR report is stored in the MySQL database. Now let us consider how SR tracks the changes to the VM. Note that SAR detects instantiation and termination of processes and generates a report. The reports generated by the SAR are compared with the information stored in the MySQL database. If there are any new processes detected in the report then a new entry is created in the database. All the new processes are tagged to be validated by the administrator and/or EV. Note that a new process could be due to the result of updates to the OS or applications, or due to a new application installed in the virtual machine or as a result of the compromise of the virtual machine. So we do not consider detection of a new processes as a proof of compromise of the virtual machine. In our architecture, the EV and/or the TVD administrator validate(s) these processes and detect(s) any malicious ones. Hence the SR captures VM specific information and tracks the changes to the OS and applications in the virtual machine

Now let us consider how the SR component enables auditing of virtual machine communications. VM communications with other hosts are logged in the database. The granularity of logging depends on the services hosted on the VM. For critical services, we log the complete traffic. For other cases, IPFIX [21] logs are reasonable. This information is used when detecting the malicious communications that were successful in generating attacks on the virtual machine. We have already considered how ED relates VM traffic to specific processes running in the virtual machine. All this information is stored in the database. In case of attacks, the database is queried to determine the malicious entity (process) at a fine granular level.

Now let us consider how SR enables restoration of services in case of attacks on virtual machines. SR maintains a default image with known good state of the virtual machine. Recall that the IM ensures that the VM boots into a known trusted state and a snapshot is taken before performing any interaction. Also, the SR takes snapshots of the VM (without memory) at five-minute intervals and deletes them automatically when the age of the snap shot is greater than 60 minutes except the first snapshot taken at every hour. It also deletes hourly snapshots after 24 hours keeping the first snapshot of the day for 30 days. Hourly and daily snapshots were kept in case a manual restore is required. If the attack occurred at time T, then the TVD Master can query SR to identify a snapshot with timestamp closest but before the attack's timestamp T and restore the VM with that snapshot.

3.5 Entity Validation

The aim of EV is to detect both known attacks as well as zero day attacks. It uses several techniques to detect attacks: input/output validation using whitelists and blacklists, signature based and anomaly based detection of attacks. It validates the VM traffic against known attack signatures and anomaly based detection policies. Note SR has specific information (such as resources allocated to VM, OS and applications running in VM, and VM traffic logs) on each virtual machine. The TVD Master makes use of this information to specify the security policies for each virtual machine. For example, signatures are selected depending on the OS and applications running in the virtual machine; similarly specific behavior monitoring to detect application anomalies and/or thresholds (such as history based thresholds for total VM traffic) are determined for anomaly based detection.

The evaluation process of the EV works as follows: We assume that the access to critical files, inputs and outputs are tagged for EV monitoring. This helps to deal with different attacks such as buffer overflow, URL redirection and injection attacks. Also all the virtual machine traffic is validated with signature based and anomaly based modules for attack detection. If any packets from a virtual machine match with a known attack signature or found to be malicious via anomaly detection, then the virtual machine is isolated from the TVD and an alert is generated to the TVD Master. The TVD Master queries the SR to determine the entity that generated the malicious traffic. For example, if the malicious entity is a new process then the recent snapshot image without the malicious entity is used for restoring the service of the virtual machine. Since attacks are possible within the threshold, we would need further offline analysis to detect the compromise of virtual machines. As part of our work, we have investigated offline analysis to detect bots which we will discuss later in Section 3.5.3. First we describe the detection of various attacks by the EV using some example scenarios.

3.5.1 Zero Day Attacks: In the case of zero day attacks, EV makes use of the following events: inputs varying from whitelists, URL redirections to blacklisted domains, unauthorized access of files that are marked sensitive (such as password files), detection of hidden process in the VM, failure to detect processes related to host based security tool in the VM, abnormal consumption of resources by the virtual machine such as CPU or memory or read/writes to the storage, packets with different properties such as spoofed source address, initiation of multiple connections and failure to establish connections, frequent termination of the established connections. For example, if a zero day attack is successful in invoking a malicious hidden process or disabling the processes related to the host based security tool in the VM, it will be detected in the SAR report; if a zero day attack is successful in exploiting vulnerability in the legitimate process/application and used for generating traffic with spoofed traffic or unauthorized ports, then it will be detected by the EV; if the attacker has disabled the auto updates to the security tools or OS in the VM, then this is detected by the anomaly detection module in EV. For example, a default Windows XP checks for updates every day at 3:00 am and Sophos security tools check for updates every 10 minutes; if malicious files are transferred with correct source address and on authorized ports, then the victim can report the malicious communications; if the attack involves altering logs related to the malicious communications in the VM, the correct logs related to the communication are still available in the SR logs and reports; we use these SR logs and reports to detect malicious communications of the VM.

3.5.2 Signature and Anomaly based Detection Attacks: Let us now consider how signature based and anomaly based modules had been developed for the EV. We are using snort attack signatures for different applications such as web server and DNS server for different attacks such as DoS, DDoS, injection attacks and spread of worms. We have also captured the behaviour of different types of malware by running them in controlled environment and characterized the attacks using Malware Attribute Enumeration and Characterization (MAEC) framework [7].

MAEC is an initiative developed by Mitre Corporation with the support of Department of Homeland Security (DHS) and National Institute of Science and Technology (NIST) [7]. One of the main aims of MAEC is to develop techniques to provide uniformity in malware reporting. This enables malware analysis results to be standardized and thus is able to support the development of databases for storing malware characteristics. MAEC attempts to use malware characteristics to identify malware based on its distinct attributes instead of single signatures used in traditional security software. MAEC also aims to accurately identify new instances or variants of existing malware families using a minimum set of malware attributes that is sufficient of characterizing the malware family.

We have captured 500 malware samples from different sources (such as malwaredomainlist and threatexpert) and analysed them in a controlled environment (e.g. Virtual Machine application like virtual box, VMWare and Xen) to capture their behaviour. The aim of performing malware analysis on this set of different samples was to gather a small size of malware attributes and behaviour and establish the behaviours and attributes that are observed as belonging to malware. We have used dynamic and static analysis techniques to determine the behaviour of the malware. Dynamic analysis only reports the runtime behaviour of the malware. It will not report the complete behaviour such as behaviour based on conditional events and when specific (input) commands need to be received from the attacker. Hence we have also used static analysis to analyse the overall behaviour of the malware. This enabled us to analyse the code and understand the inner workings of malware using reverse engineering techniques. By doing such an analysis, one can develop a graph such as the shown in Figure 5 for the malware. The figure shows a simple representation of common malware actions and the objects of these actions. The objects upon which the malware performs the actions are: file, registry keys, ports, directories, processes, mutex and services.

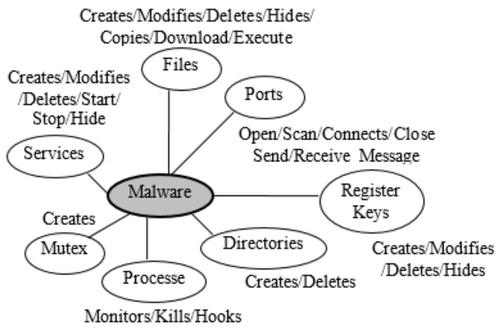


Fig. 5. Common Malware Attributes

We can develop detection based on any of these behaviours. For example, it is possible to monitor mutex using introspection since tools such as volatility have plugins to enable mutex monitoring. Hence it is easy to detect Poison Ivy backdoor since it use the default name ")!VoqA.I4"; for the mutex. Similarly Sality creates a mutex named “uxJLpe1m”, which is unique across all the variants and also uses this mutex to prevent multiple infections for the same machine. However our design choice was not to consider monitoring mutex since there can be several mutex without names and most importantly they cannot be related to any of the processes. To keep it simple, we have opted for the design choice of monitoring the processes and the traffic generated by the processes for malware detection.

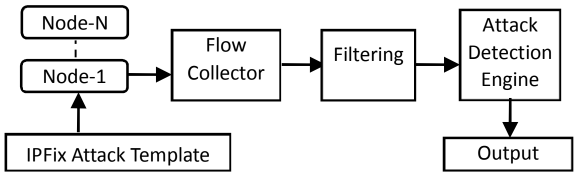


Fig.6: Offline Traffic Analysis for Detecting Attacks

3.5.3 Offline Anomaly Detection Analysis: Let us consider how offline traffic analysis is used in the detection of anomalies within the thresholds. As part of our work, we have carried out offline analysis of the TVD traffic from multiple hosts (in different clouds) to detect bots. Figure 6 illustrates a high-level overview of the bot detection system. The traffic analysis system consists of attack templates, flow collector, filtering and botnet detection engines. Attack templates are used for capturing flow information using IPFIX [21]. Flow collector is a centralised server that is used for storing and organising the data captured at different devices using the attack templates. Filtering is used to reduce the dataset, filtering out unwanted data that is not related to botnets. Finally, the botnet detection engine correlates flow information using machine learning techniques to find the patterns and detect the bots. In our analysis, we have deployed naïve Bayesian, SVM, Neural Network and Decision Trees machine learning algorithms in our analysis.

We infected some machines in our test bed environment with bot nets. Then we captured the traffic flows and identified specific features to detect the bots using machine learning algorithms. We repeated the process for different bot families. For example, Zeus bots send updates at regular interval of 20 minutes; queries and updates in peer-to-peer bots such as Nugache and Weledac create many small uniformed packets compared to legitimate P2P communications. Also the initial exchanges of packets have a fixed format and can be easily differentiated from legitimate traffic.

After analyzing a range of bots, we formulated an attack template that combines the features of network flows for detecting different bot families. We made use of the IPFIX standard to define the attack template. Different vendors have different proprietary protocols such as Cisco-NetFlow, Juniper-j-flow or cflowd, Huawei-Netstream for capturing the flow traffic. Hence by defining the uniform attack template using IPFIX, one can apply our template on devices belonging to different vendors including the virtual networking devices in the VMM. We will discuss the results from this work later in Section 4.

3.6 Virtual Domain

The Virtual Domain (VD) component is used to validate the VM traffic against the domain policies. For example, these policies specify whether communications are permitted between two virtual machines within a TVD, between TVDs and what security services need to be provided. For instance, whether encryption based confidentiality service is required for securing communications within TVDs (hosted on the same VMM, hosted on different VMM in single cloud and hosted on VMM in different clouds) and between TVDs. The TVD Master determines the security policies that need to be enforced. Communication between the domains in different TVDs is permitted according to the bilateral agreement between the TVD Masters.

As mentioned earlier, we have made use of the SDE framework [14, 22] for securing the communication within the TVD. Although SDE provides different security services such as authentication, confidentiality and integrity, it does not prevent compromised host from generating attacks on other hosts. Hence the need for the other components in our architecture such as EV and RG. We have already described how EV proactively prevents malicious communications from compromised virtual machines. Later in Section 3.8, we consider how RG enables to deal with the attacks in a reactive manner.

Now let us consider the communication between the domains. There can be a range of security policies in the TVDs. In our architecture, we have implemented information flow based security policies between TVDs. For instance, assume there are 3 TVDs: TVD1, TVD2 and TVD3. The information flow security policy specifies the flows that are allowed from and to which virtual machines in different TVDs. For instance, flows between TVD1 and TVD2 are governed by Policy12, and flows between TVD2 and TVD3 are governed by Policy23. Policy12 may state that all flows from virtual machines in TVD1 to virtual machines in TVD2 need to be protected for both confidentiality and integrity. Note that in general Policy12 consists of two sub-policies dealing with inbound and outbound flows. Inbound flows are enforced by the recipient TVD2 and the outbound flows are enforced by the sender TVD1. Hence the policies enforced with outbound flows help to deal with attacks at the source virtual machine.

Note that a VM can belong to multiple TVDs. Hence the domain policies that are to be enforced on the VM traffic are determined from the source and destination address of the VM traffic. So continuing our previous example of an auditor's virtual machine belonging to multiple TVDs, assume that the auditor wants to obtain the records from the virtual machines in TVD1. The auditor's virtual machine sends a request to TVD1 for obtaining the specific records. As the source and destination addresses of the auditor's traffic belong to TVD1, TVD1 polices are enforced on this traffic.

If an entity is continuously generating traffic violating TVD policies, then an alert is generated to the TVD Master. The TVD Master performs further validation by querying the SR database to identify the entity that is causing the violation. For example, with zero day attacks and outbreak of worms, the compromised hosts may randomly scan different IP addresses for finding the vulnerable machines and spreading the attacks. Such traffic will violate the TVD policies between the domains.

Note that VD policies are specific to each domain. For example, an organisation may decide to enforce that its staff should not communicate confidential data to other domains belonging to competitors. Even within a TVD, specific departments can have access to restricted data. For example, client machines (with student logins) in a university domain are not allowed access to a payroll server. Hence all communications of hosts within the TVD and between the TVDs need to be screened for such domain wide policies. In some cases there can also be specific policies related to release of annual reports to shareholders before making them public. Such policies are also enforced by the VD component.

Note such VD level policies are sometimes useful in the detection of insider attacks. For example, insiders transferring confidential information to competitors' domains are difficult to detect by the EV component since all the activities that a user (and the associated applications in the VM) exhibit are likely to be legitimate (e.g. no malicious user login or unauthorised access or malicious processes). It is just the organisation policy (captured by the VD) has been violated by the insider.

3.7 Local Key Management

The TVD Master is trusted for establishing the keys for secure communication of virtual machines within the TVD and between TVDs. Recall in our architecture the TVD Master and SSMs have public key – private key pairs. The KM component in the TVD Master is used for establishing the keys with the LKM components in the SSM. We use a public key based protocol similar to X.509 [23] to establish secret session key used for communications between virtual machines in a TVD. The secret session key is stored in the LKM of the SSM. The virtual machines belonging to the TVD use this key for all the interactions with other virtual machines in the same TVD whether they are hosted on same platform or on different physical paltforms/clouds.

Note if a platform is hosting virtual machines belonging to more than one TVD, then different SSMs for these different TVDs will have different secret session keys VM communications. Hence the communications of one TVD are protected from another TVD's communications. Also, since the keys are securely stored in the SSM, compromise of a virtual machine does not compromise the keys.

After the SSM validates the traffic for attacks, it is forwarded to the destination. If the destination virtual machine is hosted on the same platform, then the secret key (shared with the destination VM) is used for encrypting the traffic and the traffic forwarded to the destination. If the destination virtual machine is hosted on another physical platform, then the SDE [22] is used for securing the intra domain communication. As already mentioned, the TIB at the TVD Master maintains all the required information for securing the communications within the TVD. If the destination virtual machine belongs to a different domain, then the traffic is processed according to the bilateral agreement of security policies between the domains as specified in the TVD Masters.

3.8 Report Generation

This component is used for generating report on VM communications. The report includes information such as time, type of communication, total number of bytes and packets transferred between the source and the destination. Consider the situation where an attacker has successfully exploited a vulnerability in the VM and used the compromised machine for performing malicious actions such as forwarding files with malware and altered the logs related to the communications in the virtual machine. With our architecture, the communication details are still available in the SR logs. Note that the attacker does not have access to alter the logs in the SR. The reports generated by the RG are signed with the private key of local SSM. Hence the reports provide assurance of the virtual machine communications even when the attacker has altered the logs related to the communication. The TVD Master can therefore reliably determine whether or not a virtual machine has been compromised. Though the local SSM signs the report, in our architecture, the TVD Master further certifies the report thereby providing the assurance to the customers which helps to retain customers' trust.

Consider a scenario where a client VM in a TVD has requested downloading its bank statement from a bank server in another TVD. During such file transfer communications, there could be some additional attack traffic (not related to the bank statement) which may compromise the virtual machines. This is a classic scenario with botnets. The users are not always aware of the compromise of their machines. In this scenario, the assurance about the actual banking transaction may not provide assurance about the complete traffic, that is, all the packets transferred between the client and server VM. For example, if the client VM detects that the bank statement is being downloaded from an unpatched SQL bank server, then the client virtual machine can send malicious UDP traffic [24] to compromise the server and use it to further spread the slammer attack. Hence it is necessary to get assurance about the actual packets transferred between the client and server VM. With our architecture, we are able to achieve assurance of both coarse and finer level properties of the communications.

4 Implementation and Analysis

The configuration of our implementation is shown in Figure 7. We have used Xen VMM [5] with privileged domain running Centos. The privileged domain or Dom 0 manages the actual device by managing the complexities of the device and exports to the guest virtual machines a generic class of device driver interface for the guest virtual machines. The front end drivers are installed in the virtual machines and the back end drivers is installed in the Dom 0 which has privileged access to the physical resources. In our implementation, the SSM components are placed between the front end drivers of the guest virtual machines and the back end driver of the host virtual machine. Hence we are able to monitor the usage of resources allocated to a virtual machine and detect the attacks. Also, the monitoring components are isolated from attacks in the virtual machine since they are placed outside the VM. More details on the TVD implementation can be found in [10-12].

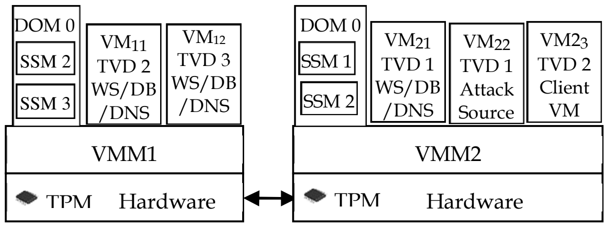


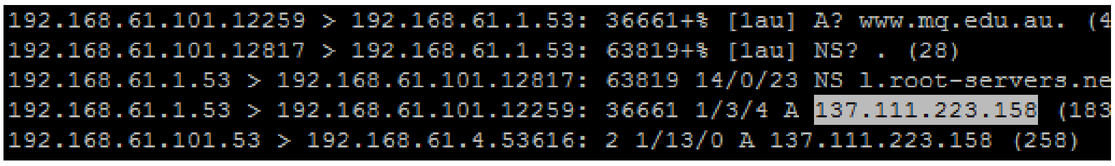
Fig. 7. Implementation on Xen

The overhead varies depending on different factors such as the services hosted in the virtual machine, applications running in the virtual machine and for different types of attacks. The default operation (relating the traffic to process + signature matching + anomaly detection) is 2%. The current attacks exhibit several malicious behaviours and our model is based on multi-level attack detection. There can be false positives for specific techniques such as for the process validation, or for anomaly detection or flow analysis techniques. However since we are using multi-level attack detection, we are able to better eliminate the false positives. For example, there can be false positives for the cases of legitimate flooding to be misclassified as attack events. Our model has second level rules for such cases. For example, if the traffic from VM exceeds threshold, then we validate the process or application that generated the traffic (such as traffic generated by well known process or traffic generated by hidden process); validate if the traffic has correct source address, and if traffic is directed to any of the blacklists. We drop the traffic only if it is generated by hidden process or if it has invalid source address or if it directed to blacklisted websites. In other cases, the traffic will be rate limited to the threshold. Hence legitimate cases will only experience in some delays.

As shown in Figure 7, virtual machines that belong to different TVDs were implemented on each physical server. Now consider a scenario where some services, such as Web service, database service and DNS service, are hosted on virtual machines; these services are accessed by the client virtual machines within the TVD as well as by the virtual machines in other TVDs according to the security policies specified in Rules (1) to (4).

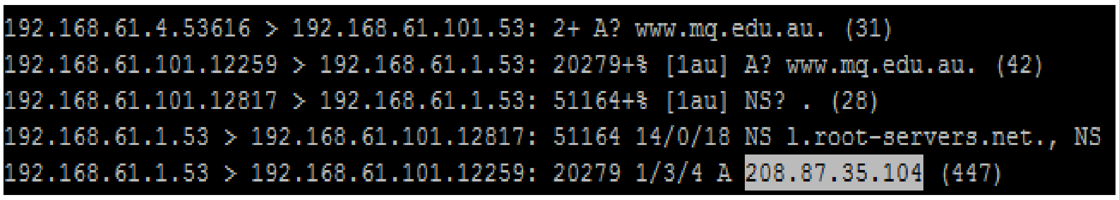
TVD1: $VM_X \leftrightarrow VM_Y$ --- (1)
TVD1, TVD2: TVD1 (VM_X) $\rightarrow$ TVD2 (VM_Y) --- (2)
TVD2, TVD3: TVD3 (VM_X) $\rightarrow$ TVD2 (VM_Y) --- (3)
TVD1, TVD3: TVD1 (VM_X) $\rightarrow | \leftarrow$ TVD3 (VM_Y) --- (4)

Rule (1) represents free communication between the virtual machines in TVD1. Rules (2) and (3) are similar to the state based security rules enforced in traditional firewalls. Rule (2) states that only the virtual machine in TVD1 can initiate the communication between TVD1 and TVD2. Similarly, Rule (3) states that only the virtual machine in TVD3 can initiate the communication between TVD2 and TVD3. Rule (4) states that no communication



```
192.168.61.101.12259 > 192.168.61.1.53: 36661+% [1au] A? www.mq.edu.au. (4
192.168.61.101.12817 > 192.168.61.1.53: 63819+% [1au] NS? . (28)
192.168.61.1.53 > 192.168.61.101.12817: 63819 14/0/23 NS l.root-servers.ne
192.168.61.1.53 > 192.168.61.101.12259: 36661 1/3/4 A 137.111.223.158 (183
192.168.61.101.53 > 192.168.61.4.53616: 2 1/13/0 A 137.111.223.158 (258)
```

Figure 8a: Authoritative DNS Resolution before compromise



```
192.168.61.4.53616 > 192.168.61.101.53: 2+ A? www.mq.edu.au. (31)
192.168.61.101.12259 > 192.168.61.1.53: 20279+% [1au] A? www.mq.edu.au. (42)
192.168.61.101.12817 > 192.168.61.1.53: 51164+% [1au] NS? . (28)
192.168.61.1.53 > 192.168.61.101.12817: 51164 14/0/18 NS l.root-servers.net., NS
192.168.61.1.53 > 192.168.61.101.12259: 20279 1/3/4 A 208.87.35.104 (447)
```

Figure 8b: Authoritative DNS Resolution after compromise

is permitted between the virtual machines in TVD1 and TVD3. As mentioned before, the TVD security policies are enforced for inbound and outbound communications of the virtual machines.

We now consider some specific attack scenarios and discuss how our architecture is able to deal with such attacks.

4.1 VM Attack Detection

First consider an attack scenario on a virtual machine that is used as an authoritative domain naming system (DNS) for the domain. DNS is one of the important services in the Internet infrastructure and is a critical requirement for all the TVDs. The primary purpose of a DNS is to translate Internet or private network domain and host names to IP addresses. In this case, we also uploaded the local resolution information into the EV component and used this as reference for trusted resolution.

DNS is vulnerable to different types of attacks. Consider the situation where the DNS server has been compromised, and a rootkit has been installed altering the local resolution information to the attacker's machine. We infected the DNS server of the TVD1 (which we called as Macquarie Domain www.mq.edu.au). In our implementation the DNS server has Avira anti-virus security software. Figure 8(a) shows the resolution before the compromise of the VM. The DNS server has been infected; the infection disables the Avira security tool [13] running in the server, alters the MQ resolution information, installs backdoor, and alters the tools (top and ls) in the VM so as not to report the malicious process and files related to the backdoor.

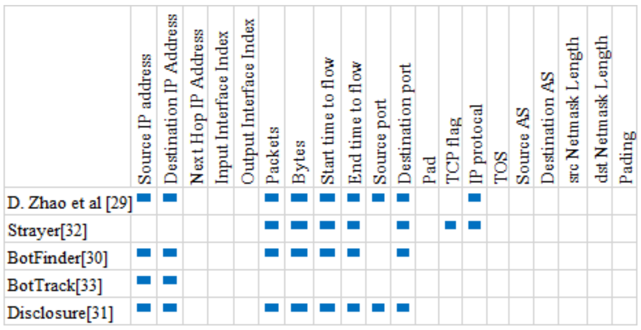
Figure 8(b) shows the resolution from the compromised DNS server. Disabling Avira and installation of the backdoor have resulted in the variation of the processes in the DNS server. Although attacker has altered the VM tools, the attacker does not have access to the SAR, since it is in Dom 0 (outside the infected VM). Hence SAR generates the correct state report of the DNS server which does not include the 3 processes related to the host based security tool and includes new process of the backdoor. Hence now we see there is a total of 60 (62-3+1=60) processes in the SAR report. Recall that the reference clean state installation has 62 processes. The new process related to the backdoor is tagged for validation by the TVD Master and/or EV. Since EV has the details of the correct resolution, it is able to determine the false resolution and drop the traffic. Since the process 967 did not match with any of the default processes and the processes related to host based security tool are missing in the SAR report, the EV raised an alert to the TVD Master and triggerd the SR to restore the DNS server with a known good image of the VM state; in this case, the complete VM is restored since it has been compromised.

Note that such attacks cannot be detected efficiently with traditional host based and network based security tools. The host based security tool was compromised by the attacker. The network based security tools can detect the false resolution but cannot detect the compromise of DNS server since they do not have the state level information of the DNS server. For example, if the attacker misuses the compromised DNS server for other malicious purpose instead of false resolutions, the network based security tools cannot detect the compromise of the DNS server. Hence we can see that our architecture is able to detect the compromise of the DNS server and restore the services.

4.2 Botnet Detection

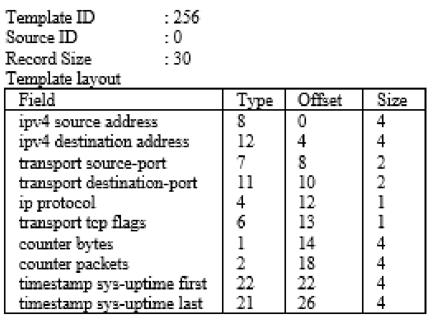
In this section, we briefly describe how offline flow analysis has been used to detect the compromise of virtual machines and their use as bots.

As mentioned earlier, first we developed a template for capturing the traffic flows for bot detection. We analysed previously proposed flow analysis techniques for detecting HTTP, IRC and P2P botnets. The related techniques make use of Netflow 5 template for capturing the flows and detecting the bots. Since NetFlow v5 comes with fixed dataset format, its record size has fixed size of 48 bytes (384 bits). Figure 9(a) illustrates the summary of NetFlow features, which were used in related techniques for detection of HTTP, IRC and P2P bot families. This clearly showed that attributes such as next hop, input, output, pad, tos, src_as, dst_as, src_mask, dst_mask and pad2 were not being used. So we developed an IPFIX template, making use of only the relevant attributes and removing the unused attributes, thereby significantly reducing the dataset size. Figure 9 (b) shows the IPFIX template with the required attributes for detecting HTTP, IRC and P2P bot families. As record size of our template is 30 bytes (240bits), there has been a 37% reduction in dataset. The reduced dataset not only reduces the corresponding storage but also helps to reduce the computations required to filter the unnecessary attributes from the data, thereby increasing efficiency of both the collector and the exporter. Also, since the captured dataset includes all the attributes of related technqiues, respective technqiues are applied on the dataset for bot detection. We make use of multi stage filtering [32] to eliminate traffic that is not realted to the attack and relevant machine learning for the attack. For example, Bayesian algorithms are used for IRC and



	Source IP address	Destination IP Address	Next Hop IP Address	Input Interface Index	Output Interface Index	Packets	Bytes	Start time to flow	End time to flow	Source port	Destination port	Pad	TCP flag	IP protocal	TOS	Source AS	Destination AS	src Netmask Length	dst Netmask Length	Pading
D. Zhao et al [29]	■	■				■	■	■	■	■	■			■						
Strayer[32]						■	■	■	■		■		■	■						
BotFinder[30]	■	■				■	■	■	■		■									
BotTrack[33]	■	■																		
Disclosure[31]	■	■				■	■	■	■	■	■									

Fig. 9(a). Attributes used in related work



Template ID	: 256
Source ID	: 0
Record Size	: 30
Template layout	

Field	Type	Offset	Size
ipv4 source address	8	0	4
ipv4 destination address	12	4	4
transport source-port	7	8	2
transport destination-port	11	10	2
ip protocol	4	12	1
transport tcp flags	6	13	1
counter bytes	1	14	4
counter packets	2	18	4
timestamp sys-uptime first	22	22	4
timestamp sys-uptime last	21	26	4

Fig. 9(b). IPFix Attack template with required attributes

decision tree algorithms are used for peer to peer bot detection. Note one can customize the attributes in IPFIX and relate them to the payload to detect other attacks such as DNS flux and spam.

The captured VM traffic flows are clustered to identify similarities in the flows. We have experimented with Bayesian, SVM, Neural Networks and Decision Trees based machine learning techniques to identify the specific features of the bots. There are three main behavioural patterns; bot behaviour, botnet behaviour and temporal behaviour. In the bot behaviour, we analysed flows generated from one bot or a single machine to identify its Command and Communications (C&C). In botnet behaviour, we analysed flows generated by group of bots or machines, in order to detect botnet activities. We horizontally analysed the flows generated in a network to find suspicious pattern related to botnet communications. Finally, in temporal or vertical method, we analysed flows generated by bots or botnets over a period of time to detect patterns.

Our analysis mainly focused on detecting the bots during different phases in botnet life cycle; initial infection, secondary infection, connection, malicious command and control, and update and maintenance. In the initial infection phase, attacker exploits vulnerabilities on victims and gains basic control over victims. The secondary infection phase is used to further download and install malicious script and binaries to get full control of the victim. Once secondary infection is complete, bots make connection to its C&C server in order to become a member of botnet. Then the bot will receive command and control from C&C server to conduct malicious coordinated activities. Finally bots update its binaries to get more functionality or evade detection.

In the case of direct C&C related bots such as IRC and HTTP, every bot needs to find its own C&C controller to be a member of centralized botnet. In the case of peer-to-peer and Hybrid bots such as Kademila, Chord and GameOver (Zeus v3) each bot needs to find its servant bot or proxy bot to get C&C instruction and become a member of P2P botnet. Our analysis confirmed that these botnets are using hard coded static IP lists or/and DNS service to locate its C&C server, from which it has to receive the control commands and updates. This generated a pattern, which is used in the detection of bots.

Even though botnets try to randomize these communications to evade detection, there are still some vertical and/or horizontal correlations in their communications. For example, we identified that the Zeus bots (v1.3) send updates at fixed interval of 20 minutes.

A main objective of the botnets is to spread the network and recruit more bots into its botnet, which ultimately benefits to surge the strength of a botnet and the attack. Hence the bots scan other machines in its network for vulnerabilities. If a vulnerable machine is found, they will run exploits to compromise the machine. When scanning, bots generate bursts of small packets. So this activity resulted in sudden increase in the number of packets (without necessarily a major increase in the traffic volume), which our detection engine tries to identify.

The attack detection engine also looked for DDoS activities such as, outbound TCP SYN packets floods and UDP floods. The reason for these large number of TCP SYN packets could mean that some of the internal hosts in the network is part of a botnet and are participating in a DDoS attack.

TABLE 2

Paper	Botnet Family			Other	
	HTTP	IRC	P2P	Size (bytes)	IP-FIX
Botfinder	Y	Y		48	
Disclosure	Y	Y		48	
Strayer		Y		48	
Zhao et al			Y	48	
BotTrack			Y	48	
Our approach	Y	Y	Y	30	Y

Table 2 compares our work with related techniques. Our architecture is able to detect a range of botnet families by examining various activities throughout the lifecycle of bots including C&C interactions, recruiting new bot members and synchronized attacks. Furthermore, since IPFix enables to capture the flows from the payload information of the IP traffic, it can also be used to detect application layer attacks. We plan to address this aspect in our future work.

4.3 Virtual Domain Attack Detection

In this section, we show how Virtual Domain enforces the TVD polices and raises alerts when the domain policies are violated. We use bots behaviour to demonstrate the violation of VD policies and detection of attacks.

As mentioned above, the bots try to spread the attacks to multiple hosts. For example, the infected machines try to scan for other vulnerable machines using random destination addresses. If the scan reports a vulnerable system, then a malicious payload is sent to infect the vulnerable host. Then the infected machine is configured to download different attacks as well as further spread the infection to other machines. We used nmap tool in the attack source (see Figure 7) to scan for vulnerable machines with random IP addresses. In the default mode, nmap scans for the well known 1000 ports on the IP addresses. We disabled the EV component at the source for this scenario. Let us consider how the VD is able to detect the attacks when the attack machine was scanning for IP addresses that belonged to

TVD2 and TVD3 which violated the policies in Rules (1) to (4).

In the first case, the scan traffic was destined to TVD2. However since the EV has been disabled at the source virtual machine, the traffic was forwarded to the VD for validation of the domain policies. According to Rule (2), communication is permitted between the TVD1 and TVD2 if the communication had been initiated from a VM in TVD1. In this case since the attack VM in TVD1 initiated the traffic, it is considered to be legitimate and the traffic is forwarded to TVD2. Similarly, the destination VD in TVD2 validates the traffic and forwards the scan traffic to the EV (Note: EV is not disabled at the destination). In this case, the EV detects the attack, drops the attack traffic and raises an alert to the TVD2 Master.

The scan traffic is also destined to TVD3. According to Rule (4), no communication is permitted between the virtual machines in TVD1 and TVD3. Hence when the nmap was scanning the IP address which belongs to TVD3, the traffic violates the VD policies. Hence this traffic is dropped by the VD in SSM1 and an alert is raised to the TVD1 Master. Hence the violation of VD policies help to deal with the attacks at the source virtual machine.

5 Related Works

In this section, we present important existing works that are relevant to our proposed architecture in this paper. The related works fall into the following categories: security policies for attack detection in TVDs, virtual machine introspection, and botnet attack detection.

The multi tenancy feature in datacenters raises several issues with respect to security policies. The concept of Trusted Virtual Domain (TVD) was originally proposed [2, 3] to simplify user and administration interactions on large scale systems by offloading the security enforcement onto the infrastructure. The work in [10] described the development of prototype using secure hypervisor and demonstrated the feasibility of managing TVDs. [11] focussed on an implementation framework which used a combination of different networking technologies (such as Ethernet encapsulation, VLAN tagging, and VPN) and security policy enforcement to realize the abstraction of Trusted Virtual Domains. These works mainly focus on achieving coarse granular security policies between the domains. Furthermore, it is not clear as to how these high level security policies can be effectively transformed to deal with different types of attacks.

In addition to providing coarse granular security between the TVDs, our architecture is also able to enforce fine granular security for communications of virtual machines within the TVD. Our architecture makes use of the SDE framework for securing the communications within the domain over different physical VMM platforms, while it uses the SSM for securing communications of virtual machines that are hosted on the same VMM. Furthermore, we have discussed in detail how the components of SSM can detect a range of malicious behaviours and current security attacks. It uses introspection, signature based and anomaly based techniques for fine granular detection of the malicious entities.

The concept of Virtual Machine Introspection (VMI) was first proposed in [6]. The critical part of VMI is the extraction of accurate system state information from the binary data that comprises the virtual machine state. However the hypervisor has no knowledge on how this information is represented in different OSs which lead to the semantic gap problem. This complicates the task of constructing the VM state which in turn results in the inefficient detection of attacks. Several techniques have been proposed [25-27] that address the sematic gap problem. Despite these works, there are still several challenges when it comes to accurate reconstruction of the OS state. Jain et al [28] conducted a detailed analysis of the introspection techniques and highlighted the open challenges that needs to be addressed in future research. The semantic gap problem can be classified into weak semantic gap (engineering problem with weak threat model) and strong semantic gap (security under strong threat model). Most papers on introspection (including ours) focus on the weak semantic gap which is solved as an engineering problem with a weak threat model. However the attackers often exploit the strong semantic gap which is still an open problem. Our VMI makes use of whitelisting of the processes from the clean state OS and applications. However, even if the attackers are successful in exploiting the semantic gap to generate attacks, these attacks can be detected using signature based and anomaly based detection modules. Furthermore, we have the facility for the victim to report the malicious behaviour of the virtual machines.

Several traffic analysis techniques have been proposed for detecting bots [29-33]. However all these techniques make use of a fixed template (e.g. Netflow v5 template) that are often vendor specific and can only detect specific bot families. In our architecture, we have identified all the attributes from Netflow v5 template that had been previously used [29-33] for detecting different bots (IRC, HTTP and peer to peer) and eliminated the attributes that were not used by any of the techniques. Then we made use of IPfix, which offers flexibility to design the template with the required attributes for capturing the flows. This has helped to minimise the record size without compromising on the ability to detect different bot families. Furthermore, IPfix templates are vendor neutral. In addition to these advantages, note that traffic analysis is just one of the security aspects used in our architecture. This is only useful for the cases of the attacks that are not detected by other mechanisms in our architecture such as introspection, signature based and anomaly based detection.

Security groups in AWS[9] or Openstack enforce Access Control Lists are based on the TCP/IP protocol stack details. For example, the security groups allows the administrator to write a rules to permit HTTP traffic to port 80. However if the HTTP traffic is generated by a malicious process or if the VM instance is compromised and sending malicious HTTP traffic, then it is still permitted by the security groups. The security groups are vendor specific and hence not suitable for end-to-end monitoring in multi-cloud environments. Our model is based on introspection and permits to use additional monitoring using signature based and anomaly based detection. For example, our model can terminate the HTTP traffic to port 80 that is generated by malicious process in the VM. Also our model is suitable for end-to-end monitoring in multi cloud environments.

6 Conclusion

In this paper, we have proposed techniques for securing services that are hosted in a multi-tenant networked cloud infrastructures. The proposed architecture is based on trusted virtual domains and it takes into account both the security policies of the tenant domains as well as specific security policies of the virtual machines in the tenant domains. We also discuss security management policies such as secure addition and deletion of a virtual machine and the revocation of privileges associated with a virtual machine in a trusted virtual domain.We describe techniques for detecting a range of attacks such as attacks between the virtual machines within a TVD, attacks between the virtual machines in different TVDs, malicious insider attacks and attacks against specific services within a TVD. The architecture also allows forensics analysis of attacks, fine granular detection of malicious entities and mechanisms for restoration of services. Finally, we have presented the implementation of architecture using Xen and illustrated how our architecture is able to secure services in networked cloud infrastructures.

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145, September 2011.

[2] A. Bussani et al., "Trusted Virtual Domains: Secure Foundations for Business and IT Services," IBM Res. Rep., RC23792, Nov 2005.

[3] J.L. Griffin et al., "Trusted Virtual Domains: Toward secure distributed services," Proc. of the 1st IEEE Workshop on Hot Topics in System Dependability, HotDep 2005, Yokohama, Japan, Jun 2005.

[4] B. Balacheff et al., Trusted Computing Platforms - TCPA Technology in Context. Hewlett-Packard Books, 2003.

[5] P. Barham et al., "Xen and the art of virtualization", Proc. of the 19th ACM symposium on Operating systems principles, NY, USA, Oct. 2003.

[6] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection", Proc. of the Network and Distributed System Security Symposium, USA, Feb 2003.

[7] "Malware Attribute Enumeration and Characterization," [Online]. Available at: http://maec.mitre.org

[8] "Microsoft Azure," [Online]. Available: http://azure.microsoft.com/ Date Accessed: 24/08/2015

[9] "Amazon Web Services," [Online]. Available at: http://aws.amazon.com/ Date Accessed: 24/08/2015

[10] S. Berger et al., "Security for the cloud infrastructure: Trusted virtual data center implementation," IBM Journal of Research and Development, Vol. 53, no.4, pp. 560-571, Jul 2009.

[11] S. Cabuk et al., "Towards automated provisioning of secure virtualised networks," Proc. of the 14th ACM Conference on Computer and Communications Security, VA, USA, Oct. 2007.

[12] L. Catuogno et al., "Trusted Virtual Domains – Design, Implementation and Lessons Learned," Proc. of the Springer International Conference on Trusted Systems, LNCS 6163, China, Dec 2009.

[13] B. Min and V.Varadharajan, "A Novel Malware for Subversion of Self-Protection in Anti-Virus", Software: Practice and Experience, Accepted and Published online in Wiley Online Library, Feb 2015.

[14] IEEE Standards for Local and Metropolitan Area Networks: Standard for Interoperable LAN/MAN Security (SILS), Sept 1998.

[15] H. Debar, D. Curry, and B. Feinstein, The Intrusion Detection Message Exchange Format, RFC 4765, Mar. 2007.

[16] R. Sailer et al., "Design and implementation of a TCG-based integrity measurement architecture," Proc. of the 13th USENIX Security Symposium, CA, USA, Aug 2004.

[17] A. Krishna, "Techniques for Trust Enhanced Distributed Authorisation using Trusted Platforms," PhD Thesis, Macquarie University, Aug 2010.

[18] A. R. Sadeghi and C. Stuble, "Property-Based Attestation for Computing Platforms: Caring about Properties, not Mechanisms," Proc. of the ACM New Security Paradigms Workshop, Canada, 2004.

[19] CSA, "Consensus Assessments Initiative Questionnaire," [Online]. Available at: https://cloudsecurityalliance.org/research/initiatives/consensus-assessments-initiative/ Date Accessed: 24/08/2015

[20] CSA, "Security, Trust & Assurance Registry," [Online]. Available at: https://cloudsecurityalliance.org/research/initiatives/star-registry/ Date Accessed: 24/08/2015

[21] B.Claise, B.Trammell and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", rfc7011, Sept 2013.

[22] V. Varadharajan, "Securing Local Area and Metropolitan Area Networks: A Practical Approach," Proc. of the 18th National Information Systems Security Conference, USA, Oct 95.

[23] D.Cooper et al., " Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" rfc5280, May 2008,

[24] D. Moore et al., "Inside the Slammer Worm," IEEE Security and Privacy, Vol. 1, no. 4, pp. 33-39, Jul-Aug, 2003.

[25] M. Carbone et al., "Secure and robust monitoring of virtual machines through guest-assisted introspection," Proc. of the Symposium on Research in Attacks, Intrusions and Defenses, The Netherlands, Sept 2012.

[26] W. Cui et al., "Tracking rootkit footprints with a practical memory analysis system," Proc. of the USENIX security symposium, WA, Aug 2012.

[27] Y. Fu and Z. Lin, "Space traveling across vm: Automatically bridging the semantic gap in virtual machine introspection via online kernel data redirection," Proc. of the 33rd IEEE Symposium on Security & Privacy, USA, May 2012.

[28] B. Jain et al., "SoK: Introspections on Trust and the Semantic Gap", Proc. of the 35th IEEE Symposium on Security & Privacy, USA, May 2014.

[29] D. Zhao et al., "Peer to Peer Botnet Detection Based on Flow Intervals", Proc. of the IFIP Information Security and Privacy Conference, Greece, Jun 2012.

[30] F. Tegeler et al., "BotFinder: finding bots in network traffic without deep packet inspection," Proc. of the ACM Int. Conf. on Emerging networking experiments and technologies, France, Dec 2012.

[31] L. Bilge et al., "DISCLOSURE: Detecting Botnet Command and Control Servers through Large-Scale NetFlow Analysis," Proc. of the Annual Computer Security Applications Conference, Florida, USA, Dec 2012.

[32] W.T. Strayer et al., "Botnet Detection Based on Network Behavior", Advances in Information Security, Vol.36, pp 1-24, Springer, 2008.

[33] J. Francois et al., "BotTrack: Tracking Botnets Using NetFlow and PageRank", Proc. of the 10th International IFIP TC 6 Networking Conference, LNCS Vol.6641, Spain, May 2011.

Vijay Varadharajan is the Microsoft Chair Professor at Macquarie University. He is also the Director of the Advanced Cyber Security Research Centre. Vijay has published more than 360 papers in international journals and conferences. Vijay has been/is on the Editorial Board of several journals including ACM TISSEC, IEEE TDSC, IEEE TIFS, and IEEE TCC.

Udaya Tupakula is a Research Fellow at the Advanced Cyber Security Research Centre at Macquarie University. In 2006, he completed his Ph.D. under the supervision of Prof. Varadharajan of Macquarie University. Uday has 50 publications in different research areas such as network security, denial of service attacks, MANET security, and secure virtual systems.