This is the publisher version of an article from the following conference:


Access to the published version: [http://dx.doi.org/10.1109/EUC.2010.87](http://dx.doi.org/10.1109/EUC.2010.87)

Copyright: Copyright 2010 IEEE. Reprinted from EUC 2010 : Proceedings of the 8th International Conference on Embedded and Ubiquitous Computing, December 11-13, 2010, Hong Kong, China. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of Macquarie University’s products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.
Detecting Security Attacks in Trusted Virtual Domains

Udaya Kiran Tupakula  Vijay Varadharajan
Information and Networked Systems Security Research, Department of Computing
Faculty of Science, Macquarie University, Sydney, Australia
{udaya.tupakula, vijay.varadharajan}@mq.edu.au

Abstract—A trusted virtual domain (TVD) enables grouping of related virtual machines running on separate physical machines into a single network domain with a unified security policy. Since the virtual machines can be running different operating systems and applications, the attacker can generate attacks in the TVD by exploiting a single vulnerability in any of the operating systems or applications. Our aim in this paper is to consider the design choices and develop an intrusion detection architecture that would enable efficient detection and prevention of different types of attacks in such a TVD based distributed environments. The proposed architecture can capture the knowledge of the operating systems and applications at fine granular level and isolate the malicious entities that are generating the attack traffic. Our model takes into account the security policies that are specific to the virtual machine as well as security policies of the trusted virtual domains to deal with the attacks efficiently.

Keywords—Trusted Virtual Domains; Intrusion Detection Systems Architecture;

I. INTRODUCTION

A virtual machine monitor (VMM) is an additional software layer that has complete control on the physical resources and enables to run multiple operating systems on a scalable computer. A Trusted Virtual Domain (TVD) [1] enables grouping of related virtual machines running on different physical machine into a single network domain with a unified security policy. Often there is free and secure communication between the virtual machines within the TVD. Any communication between the virtual machines of different TVDs is permitted according to the security policies defined by the TVD administrator. A virtual machine can become a member of multiple TVDs using a virtual network interface card (vNIC) for each TVD. However if the attacker can exploit a single vulnerability in the operating system or applications running in any of the virtual machine of the TVD, then attacks can be performed in all the TVDs in which the vulnerable virtual machine is a member.

Recently there has been considerable research [2-8] to develop security tools that are based on virtualisation. However none of these tools takes into account the properties of the trusted virtual domains to deal with the attacks. Hence there is need to develop security techniques to deal with the attacks in trusted virtual domains.

In this paper we consider the design choices for detecting the attacks and propose a novel intrusion detection architecture to deal with the attacks on the virtual machines in trusted virtual domains. Our security architecture takes into consideration the properties that are specific to each virtual machine and the domain wide security policies in which the virtual machine is a member of. The proposed architecture has mechanisms to deal with zero day attacks which are increasingly becoming significant at present.

The paper is organized as follows. Section II considers some related work, which are relevant for this paper. In Section III, we first consider the design choices and then propose intrusion detection architecture for dealing with attacks in trusted virtual domains. Section IV discusses some performance aspects based on implementation of the proposed architecture. Finally, Section V concludes the paper.

II. RELATED WORK

In this section, we present some of the important techniques that are related to our proposed architecture.

Recently some of the security tools [2-8] have been proposed to deal with attacks targeting virtual machines. ReVirt [2] securely logs information of virtual machines such as real time clock, keyboard, mouse events, user inputs and system calls by placing the logging tool in VMM. This enables the administrator to replay the logged information and analyse the attacks in case of compromise of the virtual machines. Garfinkel [3] proposed a Livewire intrusion detection system which makes use of the VMM to obtain the state of the virtual machine and detect in there is an ongoing attack. However Livewire can be used for passive detection only. Vigilante [4] is a collaborative approach where each host runs specific software which captures the information regarding the exploit of the worm and distributes a Self Certifying Alert (SCA) to warn other hosts regarding the spread of the worm. The end hosts can use the information in the SCA to identify if it is vulnerable to the worm and apply a host based filter to prevent the worm.

Cabuk et al [1] proposed techniques for implementing trusted virtual domains using techniques such as Ethernet encapsulation, VLAN tagging and VPN. Techniques such as sHype [9] and Shamon [10] have been proposed to enforce mandatory access control in virtual machine monitors. sHype [9] provides a reference monitor interface inside the hypervisor to enforce information flow constraints between virtual machine partitions. Shamon [10] is an extension of sHype to enforce mandatory access control across networked systems such as trusted virtual domains.

Although several security tools based on the VMM have been proposed none of the techniques considers the unique characteristics of the trusted virtual domains in networked
systems to deal with different types of attacks, which is the focus of this paper.

III. OUR APPROACH

In this section we consider generic architecture for trusted virtual domains and discuss the design choices for detecting the attacks. Then we propose techniques to deal with the attacks in the proposed architecture.

A. Architecture

Our architecture involves interconnected virtual machines that are hosted on different physical machines, put together to form trusted virtual domain (TVD) under a security policy. The interconnected virtual machines communicate information with each other; the information generated by the virtual machines (e.g. Ethernet frames) is encapsulated into IP packets and are destined to the virtual machines running on separate physical machines. A virtual machine becomes a member of multiple TVDs using a virtual network interface card (vNIC) for each TVD.

![Figure 1. Trusted Virtual Domains Architecture](image)

Let us consider a simplified Trusted Virtual Domain (TVD) example of our architecture shown in Figure 1, where each colour represents a trusted virtual domain. As shown in Figure 1, a physical machine can host virtual machines that belong to different TVDs and the number of colours within the virtual machine represents the number of TVDs where the virtual machine is a member of. The virtual machines can freely and securely communicate within the TVD. Any interaction between the virtual machines of different TVDs or with other hosts in the Internet is permitted according to the policies defined by the TVD administrator in our architecture.

Our aim in this paper is to develop an intrusion detection architecture framework that would enable efficient detection and prevention of different types of attacks in such a TVD based distributed environments and the isolation of the malicious entities generating these attacks. We will collectively refer to different types of attacks such as worms, viruses, Trojan horses as malware. In this paper, we will assume that the VMMs are trusted and secure. Incidentally, this is a common trust assumption that is made with many VMM or Hypervisor based systems. This assumption is based on the premise that compared to operating systems, VMMs are intended to be small in size and in principle can be verified (or designed) to be secure. If this assumption is not made, the vulnerabilities in the VMM can be exploited by the attacker to attack any or all of the virtual machines that are running on the physical machine.

In general, an intrusion detection system (IDS) can be integrated into the DOM0 (e.g. in Xen) or inside the VMM (or Hypervisor), as shown in Figure 1. Though logically equivalent, practically implementation of IDS in VMM seems to offer better performance [11]. Note that in this paper, our focus is only on the intrusion detection aspects. Security aspects such as secure communication channels between virtual machines in the TVDs have been addressed in other papers such as [1, 10].

B. Design Choices

In this section, first we consider the design choices and the challenges that need to be considered for detecting the attacks in trusted virtual domains (TVDs). These requirements have been used in the design of our architecture.

- Note that in TVDs with different virtual machines, the threat is considerably higher than in a single system, as the attacker can exploit a single vulnerability in any of the operating system or applications in the virtual machines to generate different types of attacks in all of the trusted virtual domains in which the vulnerable virtual machine is a member of. The Trusted Platform Module (TPM) technology can be used to provide attestation [12] of some properties of the OS and its applications, often at set up time, but do not necessarily guarantee intrusion free operation at runtime due to dynamic changes to the system (e.g. software and system updates).

- Each trusted virtual domain (TVD) has a unified security policy that needs to be satisfied before a virtual machine can join the domain. The VMM based IDS should take into account the TVD wide security policies as well as the specific security policies of the virtual machine to detect and deal with the attacks. For example, even if multiple virtual machines running on the VMM have same operating system, they can have different configurations, different applications and different amount of resources could have been allocated to these applications by the VMMs. Hence the attack characteristics (and the attack surface) can considerably vary for each virtual machine. For example, if very few resources are allocated to a particular VM, the attack traffic threshold for that particular VM can be low. Hence there is a need to determine and define attack signatures specific to each virtual machine.

- There can be different types of attacks on the virtual machines. It is necessary to adopt a preventive or proactive approach to deal with the attacks. However, since it is not an easy task (or even feasible) to monitor...
all the systems for all the well known attacks and prevent them there is also a need for reactive techniques. Furthermore, given the dynamic nature of attacks and the sophisticated tools available, it is becoming increasingly easy to generate polymorphic attacks in the Internet [8]. Hence the architecture should have the ability to cope with dynamic attacks; in particular, it should have mechanisms to deal with zero day attacks and dynamic polymorphic attacks.

- The efficiency of the VMM based security tool to detect and prevent attacks is dependent on the knowledge of the operating system semantics and the applications running in each virtual machine. However due to continuous updates (e.g. patches) to the operating systems and applications, there is a need to keep track of these changes and update the knowledge and the behavior of applications in virtual machines. Although the VMM has control on the virtual resources and has access to the contents of different registers, there is a semantic gap between the knowledge of the VMM and virtual machines. Greater the semantic gap, greater is the probability of false positives and false negatives (false negative is where an attack is not detected and false positive is where a legitimate event is detected as attack).

- We believe it is important for the architecture to have mechanisms to gather information from multiple locations (such as source and destination), and also combining them to detect potential attacks. For example, in the case of distributed denial of service (DDoS) attacks, each compromised machine may only contribute to small amount of attack traffic and it may not be possible to detect the attack at the source. In such cases, the attacks can be detected at the destination. In case of worms, the attacks can be detected by combining the information captured from multiple machines or infected hosts.

- Attacks should be identified at a fine granular level so that only these attacks are prevented while all the other traffic is not affected. As multiple applications or processes can be running on each virtual machine, it is necessary to identify the malicious application/process dynamically and isolate only such entities.

- The time taken for the spread of malware is continuously decreasing. Hence the architecture should provide mechanisms for automatic detection of spread of malware and for automated identification of new attack signatures. Once attack signatures are identified, there should be mechanisms to enforce dynamic filters or policies. In some cases the attacks can be efficiently prevented by applying a dynamic filter at a single location whereas in other cases such as spread of malware, filters need to be placed at multiple locations. For example, there should be possibility to drop the attack packets at the attacking (source) virtual machine or at the destination (victim) virtual machine, and in the case of spread of malware, there should be mechanism for distributed filtering.

C. System Operation

Having seen the design requirements and choices, let us now consider our intrusion detection system (IDS) architecture and its operation in terms of dealing with the various attacks. We will be referring to Figure 2 which shows the architecture for the virtual machines hosted on VMM4 in Figure 1.

In Figure 2, the security policy of virtual machine refers to the policies that are specific to each virtual machine and TVD policy represents the domain wide security policies where the VM is a member of. As mentioned earlier, there is a need to capture the knowledge of the operating system and its applications as well as the updates for each virtual machine. As the knowledge increases, this increases the potential to detect greater number of attacks and reduce attacks false positives and false negatives. In addition to this, any communication to and from a virtual machine has to be monitored against the TVD’s security policy. Since each virtual machine can belong to several trusted virtual domains, the VM interactions have to be monitored against the security policies of all the TVDs in which the VM is a member of. Figure 3 shows a detailed IDS architecture for the virtual machine (VM41) in Figure 2.

We will first present a high level description of the operation of the system and then discuss each component of the architecture in detail. The malicious entity that is responsible for the attack can be a compromised virtual machine or an application or a process that is running in a virtual machine. If the entity is a virtual machine, when an attack is detected then that virtual machine is isolated from the other machines. This causes denial of service for all the legitimate applications running in that particular virtual machine. On the other hand, if the malicious entity detected is an application or a process running in the virtual machine, then only that malicious application or the process is isolated. In our architecture, the IDS can be placed in the VMM or DOM 0 to monitor all the interactions between the virtual machines. We have implemented the IDS between the front end drivers of guest VM and back end drivers of the DOM 0 and hence it monitors all intra-VM and inter-
VM communications in terms of IP packets passing through the IDS. In this paper traffic refers to a flow or a session or a packet.

Let us now consider how the IDS architecture shown in Figure 3 is used to detect the intrusions, first at the source end. Whenever a new virtual machine is installed on the VMM, the OS Library and Repository (OSLR) component has a generic view of the operating system running in the virtual machine. So for instance in the case of Windows XP, OSLR has information on Windows XP image, service pack version and Internet explorer version, driver details, resources allocated and the details of any additional applications installed on windows. As new applications are installed, the OSLR captures this knowledge when the application starts interacting with other hosts.

Virtual machine reports the entity that is generating the traffic. Packets from the virtual machines are received by the entity validation component. This component validates the IP address of the packet. If the virtual machine has a public IP address, then the entity validation component ensures that the packet generated by the virtual machine has correct IP address. If multiple virtual machines are sharing a single IP address then the entity validation module replaces the private IP address with the shared public IP address. Entity validation component then updates the OSLR component with the details of the entity that generated the packet and passes the packet to the detection and prevention engine (DPE). If the source IP address of the packet is spoofed then the entity validation component reports this to the OSLR component and to the DPE.

Note that since the source address of all the packets is validated by the entity validation component, it is not possible for the virtual machines to generate attack traffic with spoofed source address. However at this stage, it is still possible to generate attack traffic with correct source address and we will see later how this is detected and prevented in our architecture at the source IDS. Even if the attack is not detected at the source IDS, it can be detected at the destination IDS. Also, since the attack traffic has correct source address, this enables efficient traceback of the attacking source by the destination IDS.

The OSLR component updates the knowledge of the applications running in each virtual machine as the applications start interacting with the other applications. Since the OSLR (in the VMM or Host) has access to the physical resources, it is able to validate if the process or application reported by the VM actually generated the packet. If the reported process is found in the memory, then the information can be trusted and the database is updated with the details. If the reported process/application cannot be found in the memory, then this packet is treated to be suspicious. This is then reported to the zero day attack analyzer to determine if there is ongoing zero day attack. We have optimized the overhead by validating one or more packets for each flow rather than the complete flow.

One of the main functions of the DPE is to ensure that any traffic that is entering or leaving the VMM does not contain any malicious content. We have created a database in the DPE of known attack signatures; the objective is that this database will be continually updated as and when new attack signatures are discovered. We have organized the database in such a way that the VMM administrator configures the attacks per virtual machine. Note that the OSLR has the details of resources allocated and applications running on each virtual machine. The DPE uses this information to specify the attack signatures for each virtual machine.

The anomaly intrusion detection is enabled for all the virtual machines. When the CPU is idle, the anomaly based component applies the Bayesian based learning technique [13] on the OSLR data to differentiate between legitimate and suspicious behavior for each virtual machine. We are not describing this algorithm here in this paper due to space restrictions. Essentially this algorithm enables the IDS to capture the dynamic changes for each virtual machine and identify the attacks. The evaluation process of the DPE works as follows: If any of the packet(s) from the virtual machines are matching with a known attack signature, then the packet(s) are dropped. The entity that generated the attack packet(s) is identified by querying the OSLR component and the entity is isolated or subjected to further analysis using zero day attack analyzer (we will describe
this component later). If the packet(s) do not match with any of the attacks signatures but found to be suspicious by the anomaly engine then the details of the virtual machine or entity that generated the packet(s) are stored in the shared packet buffer and a copy of the packet(s) is sent to the zero day attack analyzer for further analysis. It is important to note that if a packet is found to be suspicious then the details of the virtual machine are stored in the shared buffer. Then the DPE passes all the future packets from that particular virtual machine to the zero day attack analyzer. This helps to address the metamorphism and polymorphism characteristics of the attacks.

If the packets are found to be legitimate by the signature/anomaly based detection component, then the packets are validated against the TVD security policies and passed to the destination. 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 (corresponding to the colours in Figure 1). The information flow security policy specifies the flows that are allowed from and to which virtual machines in which TVDs. For instance, flows between TVD1 and TVD2 are governed by Policy12, and flows between TVD2 and TVD3 are governed by Policy23. Policy12 will say 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 sending TVD1. If the communication does not satisfy the TVD security policies, then the packets are dropped.

The zero day attack analyzer performs further analysis on those packets which are identified to be suspicious. Most of the zero day attacks exploit the vulnerabilities in the operating system and databases by creating buffer overflows, rewriting parts of the memory and manual jump of addresses. In some cases, the attacks come from hidden processes collecting sensitive information in an unauthorized manner or generating attack traffic to random hosts in the Internet.

Our architecture provides the following mechanisms in the zero day attack analyzer to validate the suspicious traffic at the source end. First, the entity (virtual machine/application/process) that generated the suspicious traffic is determined by querying the OSLR. Since the analyzer has access to the physical resources, it can monitor all the interactions of the suspicious entity and how the subsequent packets will be generated by the suspicious entity. The attack signatures are identified by specifying the behavior of the suspicious entity or by identifying the similarities of the packets generated by the suspicious entity. For example, process validation [5] is achieved by the analyzer by obtaining the processes that are running in the virtual machine and comparing with those obtained from the VMM. The one obtained from the VMM is a trusted report whereas the one obtained from the virtual machine is untrusted. If the number of processes listed in the untrusted report and the number of process listed in the trusted report are different then there is (are) hidden process(es). Now further analysis is performed on the hidden process by observing all the interactions of the hidden process with other entities in the virtual machine. For example, if the hidden process is accessing the inputs from the keyboard, then it can be considered as collecting sensitive information of the user and sending it to the attacker without the user’s knowledge. Alternatively if the hidden process is generating some malicious/suspicious packets to one or more destination addresses then it can be considered as sending attack traffic without user’s knowledge. Now a detailed analysis of the payload of the outgoing packets is carried out such as what type of data is being sent and to which destination.

While a decision is being made by the zero day attack analyzer on the suspicious traffic, we have a choice as to either dropping the packets (pessimistic approach) or passing or rate limiting the packets (optimistic approach) based on the characteristics of the packets. Our current implementation uses rate limited transfer of packets, if the security policies which enable easy traceback (such as validate source address (src_addr) and validate hidden process (vm_hid_prp) functions are satisfied. However if the suspicious traffic exhibits serious properties such as spoofed source address, then the packets are dropped.

After the analysis, if the zero day attack analyzer identifies the suspicious packet or a packet flow to be malicious, it determines the entity that is generating the attack traffic by querying the OSLR and isolates the malicious entity. Then the analyzer develops a new attack signature based on the properties exhibited by the malicious packet/flow and updates the attack signature database in the DPE for subsequent attack detection. For example, if the virtual machine was running an un-patched SQL server and was infected (e.g. with Slammer worm [14]), then all the suspicious instances of the outgoing flow of the worm will have 376 byte string and destined to UDP port 1434. This allows the analyser to identify UDP messages with the 376 byte string and/or destination port 1434 to be identified as attack signature and update the attack signature database in the DPE. Then the zero day attack analyzer removes the details of the suspicious virtual machine from the shared packet buffer (as the new attack signature has been included in the DPE database). To minimize the overhead, the zero day analysis can be performed in offline environment by copying the image of the suspicious VM and running it in an isolated environment.

Now let us consider how the attacks are detected at the destination IDS in our architecture. At the receiving end, the packets to the virtual machines are received by the DPE. The packets are monitored against unique security policies of the destination virtual machine (such as known attack signatures and anomaly based detection), and validated against the destination TVD security policies. If the incoming traffic matches with any of the attack signatures then the packets are dropped and there is an option to notify the source IDS. If a notification is sent to source IDS, the malicious entity that generated the attack traffic is
determined by querying the OSLR and isolated from sending similar packets to the destination virtual machine.

On the other hand, if the packet does not match with any of the attack signatures but found to be suspicious, then the destination DPE enters the details of the source virtual machine and the details of the destination virtual machine into the shared packet buffer and a copy of the packet is sent to the zero day attack analyzer for further analysis.

One of the important reasons to identify the destination virtual machine as suspicious in this case is to analyze the impact of the received suspicious packet on the destination virtual machine. Hence future packets from the destination virtual machine will also be monitored by the zero day attack analyzer. In our architecture, the destination virtual machine will be considered to be “questionable” until a decision is made on the received suspect packet by the zero day attack analyzer.

As mentioned before, while a decision is being made on the suspicious packet, the packet can be either dropped, rate limited or passed to the destination virtual machine. If the zero day attack analyzer decides not to send the packet to the destination virtual machine, then it removes the details of the destination virtual machine from the shared buffer. Since the packet is not destined to the virtual machine, this will not have any impact on destination virtual machine. On the other hand if the zero day attack analyzer decides to pass the suspicious packet to the destination virtual machine then all the packets from the destination virtual machine (which received suspicious packet) will be monitored by the zero day attack analyzer. This will be useful to identify the impact of the received suspicious packet on the destination virtual machine.

To minimize the risk of crash of the destination virtual machine, the analyzer can copy the image of the destination virtual machine and perform taint analysis [6, 7, 8] in an isolated environment by passing the suspicious packet/flow to the snapshot virtual machine. Since the analyzer has access to physical resources, first it can determine the entity that is receiving the suspicious packet and then monitor how the information in the suspicious packet or flow is processed by the entity throughout execution for suspicious behavior. If the packet or flow result in crash of the virtual machine then attack signatures can be identified from the properties of the packet/flow or from the suspicious executions of the vulnerable entity within the virtual machine. Alternatively if the entity that is receiving the suspicious packet is found to be hidden process and responding to the suspicious packet by sending attack traffic then it can be considered as control command to the hidden process. Now the attack signature can be updated in the database of the DPE. In addition the attack signature can also be updated to the source IDS.

In some cases, the information at the source IDS or the destination IDS may not be enough for determining if the suspicious packet is benign or malicious. Since in our architecture, we assume that the VMMs can securely communicate with each other, the source IDS-VMM and the destination IDS-VMM can share information to make decision on the suspicious packets/flow. The type of information shared varies based on the properties exhibited by the attack/suspicious flow. For example, in the case of spread of malware, the destination IDS can identify several similar packets from different source IDSs. In such cases, the destination IDS can identify a signature for the suspicious packets and share with the other IDSs the new attack signature.

Finally we have added the Trusted Platform Module (TPM) functionality to the architecture (See Figure 1). Most of the servers, desktops and laptops are currently being shipped with the TPM chip. This enables hardware based attestation [15] allowing a platform (e.g. host) to prove to another device that its state (e.g. operating system, programs, application image, memory content and executables) are in the “valid” reference state. The measured platform state is stored in registers called Platform Configuration Registers (PCRs). During attestation, if the state of the platform in terms of the measured state (hash values) matches with the expected values (corresponding to validation certificates provided by their respective manufacturers), then this provides the basis to believe that the platform is in a “trusted” state. In our architecture, we use the TPM to attest the state of the virtual machine monitors containing the IDSs. At present, the attested state of the VMM and the virtual machines are transferred to other VMMs as part of the inter-VMM communications. Hence the recipient VMM can trust the state of the virtual machines conveyed by the sending VMM. Currently we are in the process of extending this further to include property based attestation for trusted applications running in the virtual machines. If the packets are generated by a trusted application in the VM, then the properties associated with these packets can be checked with the properties specified by the vendor (stored in the DPE). In case of mismatch in the observed properties, then this could be used to detect the presence of malware and the packets sent by these applications can be dropped or they can be further verified with the vendor. We believe this extension will produce a trust enhanced intrusion detection architecture for trusted virtual domains, which could be potentially useful to detecting dynamic malware in the Internet.

IV. IMPLEMENTATION

We have implemented the IDS architecture shown in Figure 1. We have used Xen 3.1.2 VMM and Centos 5.1 operating system virtual machine with 4GB RAM and 2.4 GHz dual core processor. The device drivers in Xen have a front end module which are implemented in the virtual machine and a back end module in the DOM 0. The guest VM send the packets using the front end drivers and the host machine sends the packets to the guest virtual machines using the back end drivers. The policy engine is placed between the front end and back end drivers. We have specified simple TVD security policies in terms of whether flows are allowed between the TVDs. The proposed architecture involves monitoring of interactions between VMs by the IDS as well as validation against the TVD security policies. These incur additional computations and hence there is a performance overhead. However the
performance overhead can be minimised depending upon the location of the source and destination virtual machines. We conducted a range of experiments for different scenarios and examined the performance overheads.

In the first case, the source and destination VMs belonged to the same TVD and were hosted on the same physical machine (for example, TVD_{Blue}(VM_{41}-VM_{42}); TVD_{Green}(VM_{42}-VM_{43}) in Figure 2). In this case, all the virtual and physical interactions between them are monitored by the same IDS and are subject to the same TVD security policies. To check the performance overhead, we ran two instances of virtual machines and measured the outgoing and incoming traffic from one virtual machine to another on the same physical host. In this case, the two virtual machines belonged to same TVD and validation was done against a fixed number of attack signatures and anomaly behaviour. On average, this had an increase in the performance overhead of the order of 4%.

In the second case, source and destination virtual machines were hosted on the same physical machine but belonged to different TVDs (for example: TVD_{Red-Green}(VM_{41}-VM_{43}) in Figure 2) and the TVD security policies permitted communication between the entities. All the interactions between the virtual machines are monitored by the IDS against the security policies of the source virtual machine, source TVD security policy, destination TVD security policy and security policies of the destination virtual machine. The average performance impact was in the order of 5%-7%.

Then we considered cases, where the source and destination virtual machines resided on different physical machines (for example: TVD_{Green}(VM_{31}-VM_{43}) in Figure 1). The communication is monitored for security policies of the source virtual machine and TVD security policy at the source IDS, and TVD security policies and security policies of the destination virtual machine at the destination IDS. We have been carrying out experiments for this scenario. Preliminary results indicate a performance overhead of around 15%-20% where 10%-13% of the overhead seems to come from the encapsulation of Ethernet frames into IP packets. Hence a major overhead can be accounted to the encapsulation of the Ethernet frames into the IP packets, and copying information between buffers in the transmission process.

V. CONCLUSION

In this paper we have considered the design choices and proposed intrusion detection architecture to deal with attacks in trusted virtual domains. The proposed architecture takes into account security policies of trusted virtual domains as well as the unique security policies of the virtual machines. The architecture has mechanisms to capture the changes in virtual machines, thereby helping to reduce semantic gap and detect dynamic attacks. The proposed architecture has been implemented and we have also discussed some performance aspects of our model.

ACKNOWLEDGMENT

The authors would like to thank Departments of the Prime Minister and Cabinet (PM&C) and Defence Signals Directorate (DSD), Australia, for their financial support of the research project on Secure Virtualization Systems. The PM&C and DSD funding should not be taken to imply endorsement of the content or conclusions of the research project.

REFERENCES


