TVDc: Managing Security in the Trusted Virtual Datacenter

Stefan Berger, Ramón Cáceres, Dimitrios Pendarakis, Reiner Sailer, Enriquillo Valdez
IBM T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA
{stefanb, caceres, dimitris, sailer, rvaldez}@us.ibm.com

Ronald Perez
IBM T. J. Watson Research Center
1101 Kitchawan Road, Route 134
Yorktown Heights, NY 10598, USA
ronpz@us.ibm.com

Wayne Schildhauer, Deepa Srinivasan
IBM Systems & Technology Group
3039 Cornwallis Road
Research Triangle Park, NC 27709, USA
{wschildh, deepas}@us.ibm.com

ABSTRACT

Virtualization technology is becoming increasingly common in datacenters, since it allows for collocation of multiple workloads, consisting of operating systems, middleware and applications, in different virtual machines (VMs) on shared physical hardware platforms. However, when coupled with the ease of VM migration, this trend increases the potential surface for security attacks. Further, the simplified management of VMs, including creation, cloning and migration, makes it imperative to monitor and guarantee the integrity of software components running within VMs.

This paper presents the IBM Trusted Virtual Datacenter (TVDc) technology developed to address the need for strong isolation and integrity guarantees, thus significantly enhancing security and systems management capabilities, in virtualized environments. It signifies the first effort to incorporate trusted computing technologies directly into virtualization and systems management software. We present and discuss various components that constitute TVDc: the Trusted Platform Module (TPM), the virtual TPM, the IBM hypervisor security architecture (sHype) and the associated systems management software.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection – *Access control, Authentication, Cryptographic controls, Information flow controls.*

General Terms: Management, Design, Security.

Keywords: Virtualization, Mandatory Access Control, Integrity, Isolation, Virtual Trusted Platform Module, Security

1. INTRODUCTION

Datacenters such as web hosting facilities are increasingly virtualized to reduce their total cost of ownership. Cost reductions are realized by sharing each hardware platform among multiple software workloads, with each workload running in its own set of virtual machines. Consolidating workloads in this fashion has become possible because ever more powerful hardware becomes increasingly underutilized when used to run a single workload. The resulting reduction in hardware acquisition costs is accompanied by perhaps more important reductions in space, power, and cooling costs.

However, along with these cost benefits come added security concerns. Workloads that share hardware must often be kept separate for a multitude of reasons. For example, government regulations may require an investment bank to maintain a strict separation between its market analysis and security underwriting departments, including their respective information processing facilities. Similarly, commercial interests may dictate that the web sites of competing businesses not have access to each other's data.

Furthermore, concerns about malicious software become especially acute in these shared hardware environments. For example, the owner of a medical services site would like to guarantee that its software has not been co-opted by another workload to expose private information. The increased sharing of hardware by multiple workloads thus gives rise to stringent security requirements in the areas of workload isolation and software integrity.

In addition to requiring new infrastructure support for isolation and integrity, virtualized datacenters introduce difficult system management challenges. Managing the security of a networked server running a single workload is already complex. This complexity is compounded when a server is shared among multiple, possibly adversarial workloads. Appropriate mechanisms are needed to allow system managers to concisely express the rules that should govern the sharing of resources among all the workloads in a datacenter. Similarly, mechanisms are needed to monitor the integrity of all these workloads.

This paper presents the Trusted Virtual Datacenter (TVDc), a security solution that addresses both infrastructure and management issues introduced by datacenter virtualization. TVDc groups virtual machines and resources that collaborate towards a common purpose into workloads called Trusted Virtual Domains (TVDs) [7]. It provides strong isolation between workloads by enforcing a Mandatory Access Control (MAC) policy throughout a datacenter. This policy defines which virtual machines can access which resources, and by extension which virtual machines can communicate with each other. TVDc also provides integrity guarantees to each workload by leveraging a hardware root of trust in each platform to determine the identity and integrity of every piece of software running on a platform. In addition, TVDc allows centralized management of the underlying isolation and integrity infrastructure.

2. TVDc INFRASTRUCTURE

The Trusted Virtual Datacenter (TVDc) is a technology designed to enhance security and management capabilities in virtualized datacenters. It is designed to provide:

- A safety net that reduces the risk of security exposures by preventing the incorrect assignment of resources to VMs, as well as preventing unintended data from leaking and malicious software from spreading from one workload to another.
- Simplified security management based on the abstraction of Trusted Virtual Domains (TVD) [7].

The TVDc technology allows for stronger data isolation and integrity guarantees than manual resource assignments and discretionary communication controls inside the workload VMs, and it offers a higher degree of confidence that workloads and data are secure.

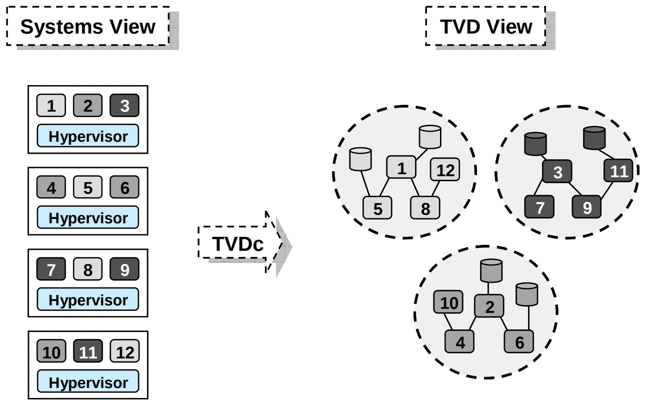

Figure 1: Trusted Virtual Domain View

Figure 1 illustrates how managing systems, virtual machines, and resources in highly dynamic virtual environments is simplified by the TVD abstraction. Without TVDc, the impact of resource assignments, migrating, cloning, suspending, and resuming VMs on workload isolation must be carefully considered in each case. VMs could migrate onto untrusted systems or could be collocated with conflicting workloads. Storage resources that have been used by one workload could be accidentally re-assigned to another workload without first being sanitized, leading to data leaks. With TVDc, each workload type, represented by VMs and resources, forms a TVD that can be managed as a whole and whose isolation and integrity properties are independent of those dynamics. Different TVDs are isolated by default and this isolation is preserved across different system architectures (e.g., Power, x86) and hypervisors (e.g., PHYP, Xen) as well as during VM lifecycle changes and resource assignments. Sharing among VMs participating in the same TVD is not restricted by the hypervisor. TVDc management ensures that resources assigned to one TVD cannot be made accessible to VMs of another TVD. A major concern for customers is integrity of each workload. The following subsections describe in more detail how TVDs are kept isolated from each other and how their integrity can be continuously monitored.

2.1 Isolation Infrastructure

Workloads are dynamically collocated in virtual datacenters to maximize availability and system utilization and to minimize power consumption and floor space. Collocating multiple workloads onto a shared physical infrastructure is essential to realize the benefits of virtualization. However, workload owners worry about isolation and confinement of their workloads when they share a common physical infrastructure. We define a workload as a set of VMs and resources that contribute to a common purpose, e.g., running a company's market analysis services.

2.1.1 Hypervisor Security Architecture

The sHype hypervisor security architecture builds on the existing isolation provided by the core hypervisor and supervises the sharing of resources among VMs as well as inter-VM communication to generalize and translate the underlying isolation of individual VMs into isolation properties of different workloads. It acts as a mediator inside the hypervisor and inside the management VM and controls access to inter-VM communication and resources according to the active security policy.

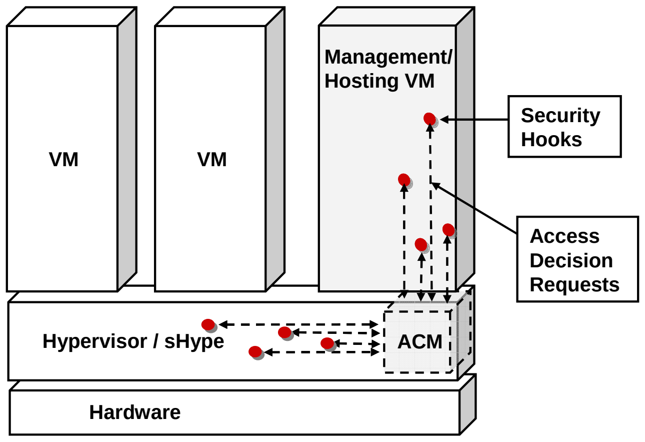

Figure 2: sHype Access Control Architecture

Figure 2 shows the sHype access control architecture as implemented for the Xen open source hypervisor. There are two sets of access mediation hooks. First, hooks controlling inter-VM communication and sharing are located inside the core hypervisor, which implements those mechanisms. In Xen, these hooks control event channel (remote interrupt) and grant tables (memory page sharing) mechanisms on top of which inter-VM communication and sharing, such as networking and virtual disk access, is implemented. Second, hooks controlling the access of VMs to resources (e.g., virtual disks, or disk partitions) are added. Those hooks are located in the management VMs; in Xen this would be domain 0. Both types of hooks call the same access decision functions inside the Access Control Module (ACM), which—in Xen—is implemented in the core hypervisor.

Moving from managing the classical individual-VM isolation to workload isolation substantially simplifies security management. No longer do administrators need to monitor individual VMs and resources but they can focus on workloads as a whole.

2.1.2 *Mandatory Access Control*

sHype guards the assignment of resources to VMs and any individual inter-VM communication by implementing mandatory access control as a reference monitor [1] inside the virtual machine monitor.

By implementing mandatory access control (MAC) in the VMM infrastructure, sHype ensures that system security policies are enforced regardless of the behavior of virtual machines. Confining different customers' workloads by mandatory access control ensures (a) that viruses and other malicious code cannot spread from one customer workload to another and (b) that data cannot easily leak from one customer workload to another even if VMs running the workloads misbehave. Additionally, it ensures (c) that break-ins or exploits in one workload do not expose other collocated workloads.

2.1.3 *Access Control Policy*

Resource assignments, resource access, and inter-VM communication are permitted or denied according to the currently active access control security policy. The security policy has two parts: (1) the label definitions, which define security contexts for VMs and resources, and (2) the anti-collocation definitions, which enforce restrictions on which VMs can run on the same system at the same time.

Security labels are composed of a reference name and a set of types. During labeling of the system, each VM and resource is assigned exactly one security label reference name. The label references are stored as part of the meta-information of VMs and resources. They are protected against the VMs and can be set and changed only through the VMM management. sHype uses those label references to determine if VMs can communicate or access resources by retrieving the label references associated with VMs and resources and permitting communication or access if those labels share a common type. This sharing model is similar to the non-hierarchical type enforcement security model [3].

Figure 3 shows how access is denied or permitted based on the security labels assigned to VMs and resources. In the figure, we differentiate between Market Analysis (MA) and Security Underwriting (SU) workloads. A trusted management virtual machine serves both workloads and exports a virtual block device that belongs to the Market Analysis workload (labeled MA). Both workload VMs can connect to the management VM since their labels share a type with the management VM. In case (1), $\{MA, SU\} \cap \{MA\} = \{MA\} \neq \emptyset$ and in case (2) $\{MA, SU\} \cap \{SU\} = \{SU\} \neq \emptyset$. Case (3) shows that the workload VMs cannot directly communicate since $\{MA\} \cap \{SU\} = \emptyset$; they share no common type. Decisions 1-3 are enforced by hooks within the hypervisor without cooperation of the management or workload VMs. The management VM (or domain 0) is trusted in turn to only connect VMs to resources, if the label of the VM and the resource share a type. This enforcement is entirely under control of the management VM, which queries the hypervisor for the label of the VM that requests to connect to a resource. It then retrieves the label of the local resource. In case (4), the VM label {SU} shares a type with the resource label {SU} of the virtual block device and the management VM grants access to mount the virtual block device. In case (5), the VM label {MA} does not share a type with the virtual block device label {SU} and this access is denied. This example also shows how the management VM preserves the workload isolation by ensuring that resources are not shared directly between different workloads. Similarly, all multi-typed VMs must be trusted not to enable illegal information flow between its types.

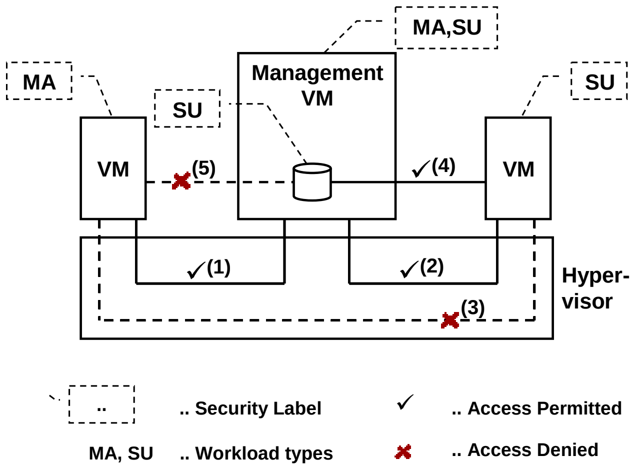

Figure 3: Controlled Sharing Decisions Based on Labels

Additionally, the security labels are used to authorize a VMM system to run a set of VMs. When a VM starts, resumes, or migrates onto the system, sHype retrieves the security label of the VMM system and the security label of the VM. If the types contained in the VMM system label are a superset of the types of the VM label, then and only then does sHype permit the VM to start. This mechanism offers a simple means to partition the physical infrastructure for stronger isolation of certain workloads.

Finally, the anti-collocation rules of the policy restrict which VMs can run simultaneously on the same VMM system. Each rule describes a set of types that conflict, i.e., only one of the types of a rule is allowed to run at a time. When a VM starts, resumes, or migrates onto a VMM system, sHype retrieves the label of the VM and derives its types. The VM is permitted to run only if the types in its label do not conflict with any type of any running VM's label. For example, to prevent different workloads from running on the same system at the same time, the conflict set {MA, SU} can be added to the security policy. Then, Market Analysis workloads (VMs and resources) must be associated with labels including the MA type and Security Underwriting workloads must be associated with labels including the SU type. sHype will then automatically enforce the collocation restriction.

sHype has been implemented for multiple hypervisors. For details on the open-source Xen implementation and a tutorial about its deployment see the Xen Users' Manual [22].

2.1.4 Network Isolation

The above has described how a TVDc-enhanced hypervisor controls VM access to resources local to the system where the hypervisor runs, for example a local disk. It is also important to enforce MAC to datacenter resources that are outside the direct control of a hypervisor, for example a network. We have thus extended our TVDc facility to control VM access to local-area networks through the use of Virtual Local-Area Networks (VLANs) that obey the IEEE 802.1Q standard [8]. Such VLANs emulate separate physical LANs on a single physical LAN by prepending to each Ethernet frame the appropriate VLAN ID and maintaining strict separation of frames with different IDs.

The basic concept behind TVDc network isolation is to associate with VLANs the same kind of label that we associate with VMs. We then restrict VMs to connect only to those VLANs with which they share a type, and to no other networks. In this way VMs that share a type can communicate through the VLAN with the same type, but cannot communicate with any other VMs.

Figure 4 shows an example of TVDc network isolation. In the figure, two physical machines (PMs) are connected through a network switch. Each physical machine is running two VMs, one labeled Market Analysis (MA) and the other Security Underwriting (SU). In addition, there are two VLANs configured throughout the system, one labeled MA and the other SU. There are two different embodiments of VLANs in the system. One embodiment is implemented in software by the VMM within each PM. The other is implemented in hardware by the network switch external to the PMs. The VMM on each PM restricts the MA VM to connect only to the MA software VLAN. Similarly, the network switch restricts each physical machine to connect only to those hardware VLANs for which the PM has an appropriate label. Finally, the VMM on each PM restricts each software VLAN to send and receive packets only on the appropriate hardware VLAN. The overall result is that MA VMs can communicate only with MA VMs, and SU VMs only with SU VMs.

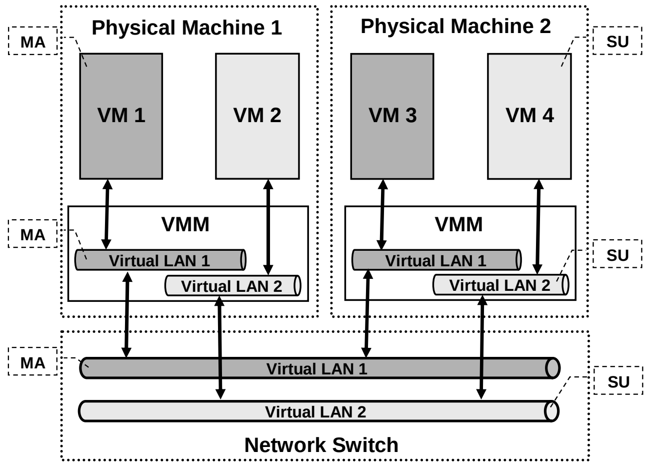

Figure 4: TVDc Network Isolation

We have prototyped TVDc network isolation using the Xen VMM and one of many commercial off-the-shelf Ethernet switches that support 802.1Q VLANs. We wrote software that configures the Xen management VM (i.e., domain 0) to connect user VMs (i.e., domUs) only to appropriate VLANs. Our software creates a software-emulated Ethernet bridge for each VLAN that a Xen system is authorized to access. When a user VM starts, our software connects it to the bridge(s) for the VLAN(s) the VM is authorized to access. In this way two VMs that share a security type on the same Xen system connect to the same software bridge and can communicate, while VMs that do not share a type cannot communicate even if they are on the same Xen system.

To extend TVDc network isolation beyond a single Xen system, our software in the management VM creates a virtual network interface with 802.1Q VLAN semantics for each VLAN the system is authorized to access, and connects that interface to the software bridge corresponding to that VLAN. Finally, we configure the external Ethernet switch to connect each Xen system only to those VLANs the system is authorized to access. In this way VMs can communicate with VMs that share a type on other Xen systems, but cannot communicate with any other VMs.

2.2 Integrity Infrastructure

One of our main objectives of the Trusted Virtual Datacenter is to provide for users and administrators as well as for software entities the ability to establish trust into software components running on different virtual machines in the datacenter. For this trust establishment to work, we can make use of software that we have previously developed for attesting to applications running on Linux on a physical machine. This software relies on the Trusted Platform Module (TPM) standardized by the Trusted Computing Group (TCG), as a hardware basis for providing an anchor for establishing a trust chain.

2.2.1 Integrity Measurement and Attestation

The TCG designed a transitive trust model architecture for the TPM device. It states that individual software components must be measured, i.e., the SHA1 hash of an executable is calculated, and its value must be extended into a register of the TPM. The procedure starts with the Core Root of Trust for Measurement (CRTM), which is code that runs early in the BIOS, that performs this type of measurement on itself and on next-to-be executed code before it transfers execution to other parts of the BIOS. For this to work, the BIOS must be instrumented with TCG extensions.

Previous work at IBM Research has extended the commonly used Grub bootloader to support the establishment of the trust chain by measuring the Grub configuration file as well as the kernel into Platform Configuration Registers (PCRs) of the TPM [13]. Further, our group developed a Linux kernel module, the Linux Integrity Measurement Architecture [18], which measures applications, libraries and device drivers launched inside the Linux operating system and extends these measurements into a dedicated register of the TPM while maintaining a log of all measurements.

A remote challenger interested in establishing trust into another system performs remote attestation to determine what software is running on the remote system. The challenged system prepares a *quote* of the current state of all PCRs along with a challenger-provided nonce and returns to the challenger the *quote* along with a list of logged measurements and the measured applications' filenames. The challenger then replays the list of hashes and simulates the PCR extend operations and compares the resulting PCR values with those provided in the *quote*. Based on the integrity of the measured applications, the challenger can determine a level of trustworthiness in the challenged system.

2.2.2 Virtual TPM

When we migrated our operating system installations from physical to virtualized platforms, we wanted to maintain the ability to attest to software running inside VMs as well as the ability to run all other TPM-aware software, such as the TrouSerS open source TCG Software Stack (TSS) developed by IBM. We achieved this by making a private TPM available to each VM on the platform by implementing TPM functionality in a software-based virtualized TPM capable of spawning multiple virtual TPM instances [2]. We placed the virtual TPM into Xen's management VM, domain 0, for easier control over it through management software in the same domain, see Figure 4. Control over the virtual TPM is necessary for the purpose of life-cycle management for each of its virtual TPM instances, such as for example spawning a new virtual TPM instance when a virtual machine requires access to a private TPM, or deleting the virtual TPM instance when the VM's configuration is removed.

Virtualization of hardware devices is typically achieved through emulation of device functionality. In case of the TPM this is an important part of how to achieve virtualization. However, a couple of challenges arise, which are directly related to the TPM being a security device. First, a TPM is usually equipped with a certificate for its endorsement key, which is issued by the device manufacturer. In the virtualized world, however, such certificates do not pre-exist for a newly created virtual TPM instance but can be created by the management software controlling the virtual TPM. Second, the transitive trust chain that is usually rooted in the CRTM in the physical machine's BIOS may now be rooted in the virtual machine's BIOS and measured into the virtual TPM instance associated with the VM. The problem with this is that an outside challenger would not see the complete trust chain that is relevant for not only establishing trust into software running inside the VM, but also into software relevant for the VM and hypervisor implementation. In this case, the certificate can indicate that a *quote* is generated by a virtual TPM and the challenger can determine to follow the additional steps and attest the underlying virtualization infrastructure.

Figure 5 shows the architecture of the virtualized system with the virtual TPM inside the system's management VM. A system management stack, based on the Common Information Model (CIM), is used to control the life cycle of virtual machines and that of virtual TPM instances. VMs that use TPM functionality only have access to only their own virtual TPM instance.

An important part of the trust establishment architecture is the verification of the hashes of software reported to be running on a system. Since there are many different pieces of software available in various versions, a very large number of hashes can be expected to exist, and along with that a scalability problem arises that requires centralized management of hash information along with annotations about the quality of the software. Annotations may for example include known vulnerabilities or even information about malicious software. By maintaining a centralized database of hashes we relieve every challenger from maintaining its own database. This model turns out to fit nicely into the concept of system management, which by its nature implements a centralized point for remotely managing virtual machines in a datacenter.

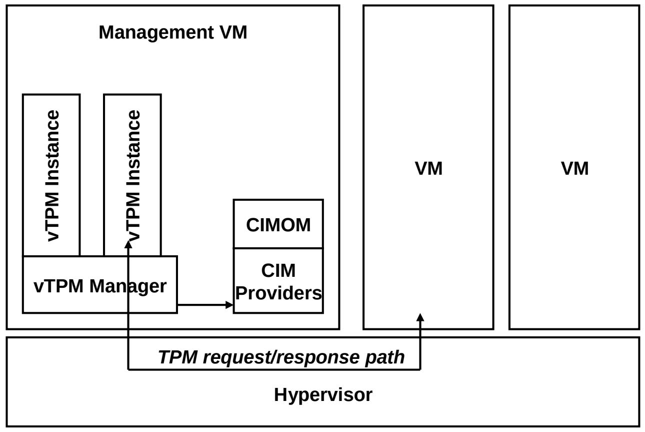

Figure 5: Virtual Trusted Platform Module Architecture

Another important aspect of TVDc is the ability of the system administrator to monitor the status and trust level of software running inside deployed virtual machines, i.e. for Integrity Management. To support this, an agent monitors measurements occurring inside a virtual machine and forwards them to the management application for continuous tracking of system health status.

3. TVDc MANAGEMENT

Critical for enabling isolation and integrity is a central point from which an administrator can simply establish and deploy security policies and consolidate measurement data for analysis. A systems management application designed to exploit the infrastructure and features described in the previous sections provides these capabilities.

Figure 6 shows a three-tiered systems management application, such as IBM Systems Director. The top block represents the user interface through which the administrator interacts with the isolation and integrity management functions provided on the systems management application server. It also presents the configuration of the datacenter and the results of the requested operations. The user interface may be a web browser, a graphical console, or a command line. The second tier is the systems management application server. The third tier includes the management VMs, which are the entities through which the systems management server controls the host environments and their virtual machines. (Please see Figure 5 for a detailed block diagram of the Management VM platform.) The systems management server communicates with its endpoints using the Common Information Model.

For isolation operations, the systems management server enables the setting and verification of isolation policies, and the creation and assignment of labels to the virtual machines and their resources. For integrity, the systems management server allows the administrator to attest the management VMs and their respective virtual machines.

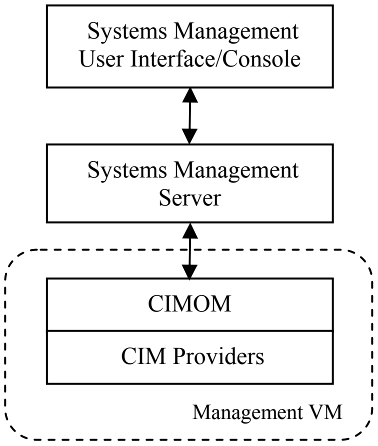
Systems Management User Interface/Console
Systems Management Server
CIMOM
CIM Providers
Management VM

Figure 6: Systems Management Application Stack

In addition to managing the isolation and integrity functions on the management VMs, the Trusted Virtual Datacenter requires the following facilities of the systems management server. First, the TVDc implies different levels of administration: administrators of the IT datacenter (physical resources) must be able to manage all the resources allocated and available for use in the physical datacenter, while the administrators of a virtual datacenter should only manage resources allocated to that virtual datacenter. Second, a TVDc should implement only one policy for all the management VMs it contains. Third, as a consequence of the first two requirements, the management VMs of the virtual datacenter should be grouped together and administered as a group. Fourth, the systems management application must provide an event, or notification, infrastructure that indicates anomalies and allows for their remediation.

3.1 Isolation Management

The systems management functions for isolation enable the administrator of the IT datacenter to group the management VMs into a TVDc and to create the policies and isolation labels necessary for the virtual machines within the group. Generally, the tasks of this administrator include:

- Policy creation, modification, deployment, and enabling
- Isolation label creation and deletion
- Remediating notifications of policy violations or attempted modifications

As shown in Figure 7, the administrator of the TVDc must have the ability to assign the isolation labels created by the overall Datacenter Administrator to virtual machines and their resources with his group. Error messages or indications must be displayed for mis-assigned or conflicting resources.

A use case for creating and configuring a TVDc follows. The overall Datacenter Administrator:

1. Discovers the management VMs within the datacenter.
2. Assigns the management VMs to a group to be managed as a TVDc.
3. *Defines* the isolation labels and exclusion rules (anti-collocation rules) for the TVDc.
4. Deploys the resulting policy from the previous step to the management VMs in the TVDc.
5. Authorizes the TVDc Administrator to view the created TVDc and assign isolation labels.

After authorization and authentication, the TVDc Administrator:

6. Uses the labels created by the overall Datacenter Administrator and assigns them to the VMs of the management VMs within the group.

Note that the TVDc Administrator cannot create or alter the isolation policy or labels.

The systems management application should allow both administrators to easily and clearly view the security attributes of the TVDc, its domains, and the resources of the domains.

For VLAN isolation, it is desirable, but not essential, for the overall Datacenter Administrator to have the ability to create new virtual interfaces and bridges necessary to assign the correct isolation labels for connecting virtual machines in a TVDc.

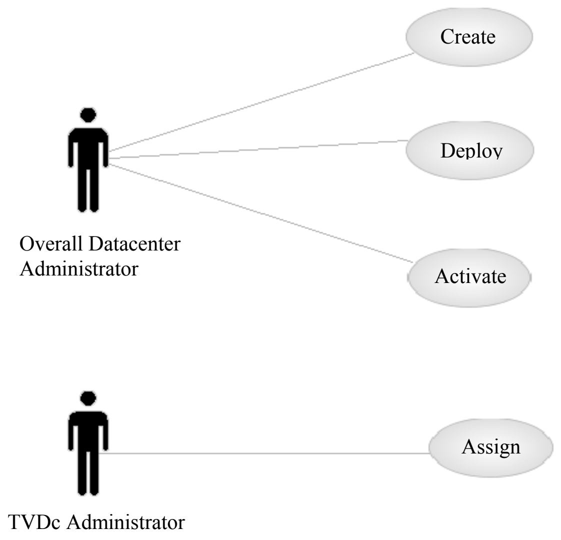

Figure 7: Administration Responsibilities for Isolation

3.2 Integrity Management

Fundamentally, the systems management application implementing integrity management performs remote attestation on a management VM or a virtual system, acting as the challenging party. In a Xen installation, it gathers the integrity measurements for domain 0 and its user domains from the virtual TPM on domain 0 via CIM and compares the received list with the database of known, trusted measurements. If an unknown or untrusted measurement is found, it generates an event to the administrator for remediation.

If the systems management application has an automated event facility, e.g. the IBM System Director Event Action Plans, the administrator can set up an action prior to an event being received. Depending on the criticality of the virtual system under test, it can be shutdown, restarted, an e-mail sent, etc.

For the trust chain to be unbroken, the system hosting the management VM can be equipped with TPM 1.2 and have static Core Root of Trust for Measurement, such as the IBM System x 3850 M2. In addition, a measured boot-loader, such as Grub with the TCG patch to support trusted boot, must be installed and the Integrity Measurement Architecture (IMA) enabled.

An important goal of the integrity management design, met by the virtual TPM, is the capability to obtain measurements for the virtual machines from the management VM to eliminate or minimize any potential agent on the virtual machines themselves. Additionally, by using the virtual TPM with the CIM providers shown in the example, the measurements can be obtained securely without requiring a *quote* containing the current PCR values signed by the virtual TPM.

The systems management application must have access to a reference measurements list of trusted components, as described in Section 2.2.2. The systems management application can be configured with a default list of measurements, but it must provide the administrator the ability to add new measurements from a trusted environment including those for a new virtual image, configuration files, updates for packages contained in the original list, etc. Likewise, the administrator must be able to delete stale or untrusted measurements from the reference list in the case of updates or security patches.

A use case for integrity management includes the following steps. The overall Datacenter Administrator:

1. Ensures that the necessary reference measurements are provided in the reference database or are imported from a trusted source.

The TVDc Administrator:

2. Identifies the criticality of the VMs to be measured and the remediation.
3. Sets an action plan on the systems management server to react to events indicating untrusted measurements. For example, the action may be to 'shutdown' an untrusted VM.
4. Enables the systems management application to monitor the VMs to be attested, either via agents or by scheduling the integrity attestation task.

4. RELATED WORK

The TVDc work integrates virtualization-based security and system management to provide trust and containment guarantees to the datacenter environment. This work is an implementation of the Trusted Virtual Domain (TVD) abstraction [5] and [6], which aims to simplify user and administration interactions on large scale systems by offloading the security enforcement onto the infrastructure. TVDc builds on previous work on trusted computing [19], virtual TPM [2], trust measurement architecture [10],[18], and sHype mandatory access control extensions to the Xen [17] and PHYP [20] hypervisors. Independently of our work, Cabuk et al. [1] developed a method that assigns VLANs to separate TVDs, which is similar to our own network isolation approach as described in Section 2.1.4.

Other related work in [11] and [12] strengthens grid security by proposing the use of TPMs to provide cryptographic credentials, remote attestation, and integrity protection. This is similar to our distributed mandatory access control work [15], which establishes verifiable trust in virtualized environments running a grid distributed application. Another ongoing work is the Open Trusted Computing [16] initiative which proposes to develop a trusted and secure computing system based on open-source software.

NetTop [14] provides similar functionality as TVDc for client systems using commercial-off-the-shelf hardware and software. NetTop leverages virtualization to replace multiple end-user workstations having different security labels with VMs on a single physical system and utilizes virtual network configurations to interconnect VMs with secure enclaves. However, the NetTop architecture relies on the security controls of the host OS, since the VMM runs on top of the underlying OS. Our approach, in contrast, provides VM access control at the lowest levels of the system.

For future work, we can leverage hardware security features such as Intel's Trusted Execution Technology [9] to establish protected environments without the need to reboot systems.

5. CONCLUSIONS AND FUTURE WORK

Our prototype has proven the feasibility of managing Trusted Virtual Domains across servers, networks and storage resources. However, several research challenges need to be addressed in order to facilitate TVDc deployment and operation. Some of these challenges that we are currently addressing are listed next.

An obstacle to the adoption of virtualization is the complexity of customer on-boarding onto a TVDc. Solutions that migrate a group of physical servers along with their interconnected networking and storage recourses in such a way as to preserve, or even improve, the security and isolation of the original configuration are needed. Such solutions will alleviate the concerns that customers legitimately have regarding potential security vulnerabilities introduced by the migration of pre-existing workloads onto a shared physical infrastructure.

The current TVDc prototype provides strong, but coarse grain isolation; this complicates the sharing of datacenter-hosted services that need to be visible to clients or between business partners. We are currently working on the design, automatic configuration and optimal placement of controlled sharing devices that could be implemented in special VMs or within the hypervisor and that can provide controlled sharing between different TVDs within a TVDc. Controlled sharing policies should be linked with higher level business process management policies. We envision the automatic configuration of these mechanisms that enable external service requests to be safely routed to the corresponding TVD while minimizing the related security exposure.

Another area of future work relates to the interaction of different management constraints, such as availability and resource management, with security constraints, such as anti-collocation rules. When these capabilities are managed independently, sub-optimal or even conflicting decisions may result. Finally an important area of future work regards the extension of our current prototype to management of a heterogeneous environment consisting of different hypervisors, in addition to Xen and PHYP, as well as different networking technologies. In such a heterogeneous environment, different APIs and isolation capabilities may be present in each hypervisor and must be coordinated to achieve the overall TVD-based isolation guarantees.

6. ACKNOWLEDGMENTS

Several people in IBM have contributed to TVDc, as well as the related secure virtualization and trusted computing efforts. We would like to especially thank William Armstrong, Steven Bade, Leendert van Doorn, Kenneth Goldman, Peter Heyrman, Stephen Levesque, José Moreira, Eran Rom, Sivan Tal, the PHYP team, the System x Virtualization team, and the Linux Technology Council (LTC) Security team. We would also like to thank the Xen community for their support.

7. REFERENCES

[1] J. P. Anderson. Computer Security Technology Planning Study. ESD-TR-73-51, Vols. I and II, Air Force Electronic Division Systems, Hanscom AFB, Bedford, MA, Oct. 1972.

[2] S. Berger, R. Cáceres, K. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the Trusted Platform Module. 15th USENIX Security Symposium, July 2006.

[3] W. E. Boebert and R. Y. Kain. A Practical Alternative to Hierarchical Integrity Policies. 8th National Computer Security Conference, 1985.

[4] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. IEEE Symposium on Security and Privacy, May 1989.

[5] A. Bussani, J. L. Griffin, B. Jasen, K. Julisch, G. Karjoth, H. Maruyama, M. Nakamura, R. Perez, M. Schunter, A. Tanner, L. van Doorn, E. V. Herreweghen, M. Waidner, S. Yoshihama. Trusted Virtual Domains: Secure Foundations for Business and IT Services. Research Report RC23792, IBM Research, November 2005.

[6] S. Cabuk, C. I. Dalton, H. Ramasamy, and M. Schunter. Towards Automated Provisioning of Secure Virtualized Networks. Research Report RZ3692. IBM Research, June 2007.

[7] J. L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn, and R. Cáceres. Trusted Virtual Domains: Toward Secure Distributed Services. 1st IEEE Workshop on Hot Topics in System Dependability, June 2005.

[8] IEEE Std. 802.1Q-2003, Virtual Bridged Local Area Networks; ISBN 0-7381-3662-X.

[9] Intel Corporation. Trusted Execution Technology Preliminary Architecture Specification, August 2007. URL:http://www.intel.com/technology/security/downloads/315168.htm

[10] T. Jaeger, R. Sailer, and U. Shankar. PRIMA: Policy-Reduced Integrity Measurement Architecture. 11th ACM Symposium on Access Control Models and Technologies (SACMAT), June 2006.

[11] W. Mao, H. Jin, and A. Martin. Innovations for Grid Security from Trusted Computing. White paper, June 2005.

[12] W. Mao, F. Yan, and C. Chen. Daonity-Grid Security with Behavior Conformity from Trusted Computing. 1st ACM Workshop on Scalable Trusted Computing (STC 2006).

[13] H. Maruyama, F. Seliger, N. Nagaratnam, T. Ebringer, S. Munetoh, S. Yoshihama, and T. Nakamura. Trusted Platform on Demand. Technical Report RT0564, IBM, February 2004R.

[14] Meushaw and D. Simard. NetTop-Commercial Technology in High Assurance Applications. National Security Agency Tech Trend Notes, Fall 2000.

[15] J. M. McCune, S. Berger, R. Cáceres, T. Jaeger, and R. Sailer. Shamon–A System for Distributed Mandatory Access Control. 22nd Annual Computer Security Applications Conference (ACSAC), December 2006.

[16] Open Trusted Computing. URL:http://www.opentc.net.

[17] R. Sailer, T. Jaeger, E. Valdez, R. Cáceres, R. Perez, S. Berger, J. L. Griffin, and L. van Doorn. Building a MAC-based Security Architecture for the Xen Opensource Hypervisor. 21st Annual Computer Security Applications Conference (ACSAC), December 2005.

[18] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. 13th USENIX Security Symposium, August 2004.

[19] Trusted Computing Group. URL:https//www.trustedcomputinggroup.org.

[20] E. Valdez, R. Sailer, and R. Perez: Retrofitting the IBM POWER Hypervisor to Support Mandatory Access Control. 23rd Annual Computer Security Applications Conference (ACSAC), December 2007 (Accepted for publication).

[21] F. Yan, W. Quang, Z. Shen, C. Chen, H. Zhang, and D. Zou. Danoity: An Experience on Enhancing Grid Security by Trusted Computing Technology. ATC, volume 4158 of LNCS, Springer, 2006.

[22] Xen Users' Guide Chapter 10 for the Xen sHype/Access Control Module: http://www.cl.cam.ac.uk/research/srg/netos/xen/readmes/user/user.html