Providing Security for Virtual Datacentres

Sas Mihindu (Editor)
Knowledge Engineering Consultant
mybox@sas.mihindu.name

Abstract

This paper provides technique for realising integrity and isolation in virtual systems. This is achieved by supporting a logical cages model, in particular for virtualised datacentres, based on a concept called Trusted Virtual Domains or TVDs [5]. Based on previous work, paper describes a security management framework that helps to realise the abstraction of TVDs by guaranteeing reliable isolation and flow control between domain boundaries. The proposed framework employs networking and storage virtualisation technologies as well as Trusted Computing for policy verification. The main contributions are (1) combining these technologies to realise TVDs and (2) orchestrating them through a management framework that automatically enforces isolation among different zones. In particular, this solution aims at automating the verification, instantiation and deployment of the appropriate security mechanisms and virtualisation technologies based on an input security model, which specifies the required level of isolation and permitted information flows.

1 Introduction to virtualised security

Hardware virtualisation is enjoying a resurgence of interest fuelled in part by its cost saving potential in datacentres. By allowing multiple virtual machines to be hosted on a single physical server, virtualisation helps improve server utilisation, reduce management and power costs, and control the problem of server sprawl. Techniques need to be developed in the security management of virtual machines, i.e., the protection, enforcement, and verification of the security of virtual machines. Security management is a non-trivial problem even in traditional non-virtualised environments. Security management of virtual machines (VMs) is even more complicated because the virtual machines hosted on a given physical server may belong to different virtual organisations, and as a result, may have differing security requirements. Protecting a VM against security attacks may be complicated by inadequate isolation of the VM from other VMs hosted on the same server. Verifying the security of a VM may be complicated by confidentiality requirements, which may dictate that the information needed for verification of a VM’s configuration should not divulge configuration information of other co-hosted VMs.

This paper addresses two main problems relating to security management, particularly integrity management, of VMs: (1) protecting the security policies of a VM against modification throughout the VM’s lifecycle, and (2) verifying that a VM is compliant with specified security requirements. This describes a formal model that generalises integrity management mechanisms based on the Trusted Platform Module (TPM) [23] to cover VMs (and their associated virtual devices) and a wider range of security policies (such as isolation policies for secure device virtualisation and migration constraints for VMs). On TPM-equipped platforms, system compliance can be evaluated by checking TPM register values. The proposed model allows finer-grained compliance checks by handling policies that can be expressed as predicates on system log entries. Verifying compliance involves showing that the system integrity state, as reflected by secure write-only logs, satisfies certain conditions. These techniques build on previous work by others [3, 12, 13] who have used the Trusted Platform Module (TPM) [23] to protect the integrity of the core virtual machine monitor (VMM) and to reliably isolate VMs. Based on the formal model, the paper describes an integrity architecture called PEV (which stands for protection, enforcement, and verification) and associated protocols. The architecture incorporates integrity protection and verification as part of the virtualisation software itself, and at the same time enhances its policy enforcement capabilities. This work has gone through proof-of-concept by dissemination of a prototype realisation of the architecture using a specific hypervisor. This work has demonstrated the policy enforcement and compliance checking capabilities of a prototype through multiple use cases. The proposed, generalised integrity management mechanisms are both extensible and flexible. *Extensibility* means that it is possible to guarantee compliance even if new virtual devices are attached to the VMs. *Flexibility* means that the verifier is able to specify which aspects of the enforced security policies are of interest, and obtain only the information corresponding to those aspects for validation of system compliance.

1.1 Security Objectives

The main security objective is to provide isolation among different domains that is comparable with the isolation obtained by providing one infrastructure for each customer. In particular, we require a security architecture that protects those system components that provide the required isolation or allow to verifiably reason about their trustworthiness of and also of any peer endpoint (local or remote) with a domain, i.e., whether they conforms to the underlying security policy. This has been achieved by grouping VMs dispersed across multiple physical resources into a *virtual zone* in which customer-specified security requirements are automatically enforced. Even if VMs are migrated (e.g. for load-balancing purposes) the logical topology reflected by the virtual domain should remain unchanged. This research deploys Trusted Computing (TC) functionalities to determine the trustworthiness (assure the integrity) of the policy enforcement components. Therefore this model would provide better flexibility, adaptability, cost savings than today's physical cages model while still providing the main security guarantees required for applications such as datacentres.

Datacentres provide computing and storage services to multiple customers. Customers are ideally given dedicated resources such as storage and physical machines. In the physical cages approach, only few resources such as the Internet connection may be shared between multiple customers. For cost efficiency, our logical cages approach promotes securely extending sharing to other resources such as storage and networks. This is enabled by preventing unauthorised information exchange via shared resources. To model and implement the logical caging approach, the research introduces a domain-based security model for enforcing unified security policies in virtualised datacentres. It focuses on isolation policies that mimic physical separation of datacentre customers. The goal is to logically separate networks, storage, VMs, users, and other virtual devices of one customer from another customer. For this purpose, it defines *domain isolation* as the ability to enforce security policies within a domain independently of other domains that may co-exist on the same infrastructure and interact with that domain. The core idea is to use this isolation property as a foundation for guaranteeing desired security properties within each virtual domain while managing shared services under mutually agreed policies.

This paper explains the policies that describe the controlled information exchange in a virtualised datacentre. In order to put this work in context, the research surveys key concepts that underlie this approach. Section 2 presents the TVD concept, which can be thought of as a virtualisation of today's security zones while making security requirements explicit. Section 2.1 describes *Trusted Computing* concepts. The core of this concept is a security hardware device called *Trusted Platform Module* that guarantees certain security functionalities in spite of attacks. This also surveys related work on trusted channels in Section 2.2 and on secure virtual networking in Section 2.3. Section 3 describes the individual components that enable this research to enforce the discussed policies and TVD-based policy enforcement framework.

2 Trusted Virtual Domains

Bussani *et al.* [5] introduced the concept of TVDs. A Trusted Virtual Domain consists of a set of distributed Virtual Processing Elements (VPEs), storage for the VPEs, and a communication medium interconnecting the VPEs [5, 13, 15]. The TVD provides a policy and containment boundary around those VPEs. VPEs within each TVD can usually exchange information freely and securely with each other. At the same time, they are sufficiently isolated from outside VPEs, including those belonging to other TVDs. Here, isolation loosely refers to the requirement that a dishonest VPE in one TVD cannot exchange information (e.g., by sending messages or by sharing storage) with a dishonest VPE in another TVD, unless the inter-TVD policies explicitly allow such an exchange. There is a TVD *infrastructure* (for each TVD) that provides a unified level of security to member VPEs, while restricting the interaction with VPEs outside the TVD to pre-specified, well-defined means only. Unified security within a virtual domain is obtained by defining and enforcing *membership requirements* that the VPEs have to satisfy before being admitted to the TVD and for retaining membership.

Each TVD defines rules regarding information exchange with the outside world, e.g., restrictions regarding in-bound and out-bound network traffic. Figure 2.1 shows customer VMs as VPEs belonging to TVD1 spanning two platforms (contained in the dashed boxes). TheMaster (TVD1 Master) and Proxy components (Proxy1 on each platform) are part of the TVD infrastructure, which is described in Section 3.3. The TVD Master is the orchestrator of the TVD deployment and configuration. There is one TVD Proxy for each platform hosting VMs belonging to that TVD. If the platform hosts VMs belonging to multiple TVDs, then there are multiple TVD proxies on that platform, one per TVD. The TVD Proxy on a platform is configured by the TVD Master and can be thought of as the local TVD policy enforcer. VMs belonging to the same TVD can usually exchange information freely with each other unless restricted by VM-level policies. For example, traffic originating from VM_{A1} or VM_{A2} on Host A is routed to VM_{Bi} (i = 1,…, 4) on Host B without any restrictions. Information exchange among TVDs can be allowed; however, it is subject to the network and storage policies stated by each TVD Master and locally enforced by each TVD Proxy.

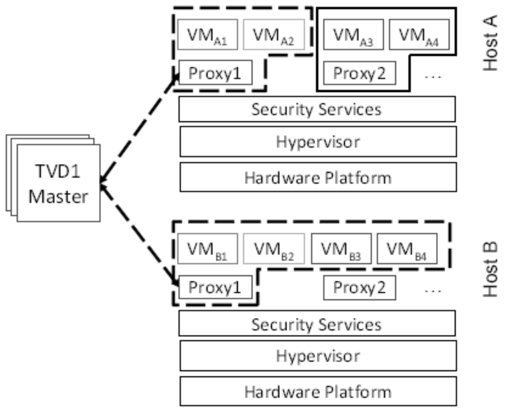


Figure 2.1: TVD Architecture: High-Level Overview [7].

2.1 Trusted Computing – The TCG Approach

It is important to have reliable mechanisms for a system to reason and verify the trustworthiness (i.e., compliance with a certain security policy) of a peer endpoint (local or remote). A recent industrial initiative towards realising such a mechanism was put forward by the *Trusted Computing Group* (TCG) [22], a consortium of a large number of IT enterprises that proposes a new generation of computing platforms that employs both supplemental hardware and software. The TCG has published several specifications on various concepts of trusted infrastructures.

The Trusted Platform Module: The core component the TCG specifies is the *Trusted Platform Module* (TPM). Currently, the widespread implementation of the TPM is a small tamper-evident chip that implements multiple *roots-of-trust* [23], e.g., the root-of-trust for reporting and the root-of-trust for storage. Each root-of-trust enables parties, both local and remote, to place trust on a TPM-equipped platform that the latter will behave as expected for the intended purpose. By definition, the parties trust each root-of-trust, and therefore it is essential that the roots-of-trust always behave as expected. Given that requirement, a hardware root-of-trust – especially one that is completely protected from software attacks and tamper-evident against physical attacks, as required by the TPM specification – is assumed to provide a better protection than software-only solutions.

Attestation and Integrity Verification: The Trusted Computing features that leverage in this paper are protection of keys, secure recording of integrity measurements, attestation, and sealing. Integrity verification mechanisms enable a remote party to verify whether system components conform to certain security policies. *Measurement* of a component involves computing the SHA-1 hash of the binary code of that component. In particular, each software component in the Trusted Computing Base (TCB) is first measured and then its measurement recorded before control is passed to it. The hash values are then appended to a hash chain, which is kept in special protected registers called *Platform Configuration Registers* (PCRs), thus acting as accumulators for measurements. *Recording* a measurement means appending it to the hash chain by PCR extend operation. The sequence of measured values are also stored in a *measurement log*, external to the TPM.
Attestation refers to the challenge-response-style cryptographic protocol for a remote party to query the recorded platform measurement values and for the platform to reliably report the requested values. The verifier first sends a challenge to the platform. The platform invokes the TPM_Quote command with the challenge as a parameter. The invocation also carries an indication of which PCRs are of interest.

The TPM returns a signed *quote* containing the challenge and the values of the specified PCRs. The TPM signs using an Attestation Identity Key (AIK), whose public key is certified by a third party that the verifier trusts, called *Privacy CA* in TCG terminology. The platform then replies to the verifier with the signed quote along with the AIK public key certificate and the log information that is necessary to reconstruct the platform’s configuration. Based on the reply, the verifier can decide whether the platform is in an acceptable state.
Sealing is a TPM operation that is used locally to ensure that a certain data item is accessible only under specific platform configurations reflected by PCR values. The *unsealing* operation will reveal the data item only if the PCR values at the time of the operation match the PCR specified values at the time of sealing.

A more general and flexible extension to the binary attestation is *property-based attestation* [17, 21]: Attestation should only determine whether a platform configuration or an application has a desired property. However, the prototype is using binary attestation. In [14], the authors propose *semantic remote attestation* using language-based trusted VM to remotely attest high-level program properties. The general idea is to use a *trusted VM* (TrustedVM) that verifies the security policy of another virtual machine on a given host. Previously authors proposed software architectures based on Linux providing attestation and binding. The architecture binds short-lifetime data (e.g., application data) to long-lifetime data (e.g., the Linux kernel) and allows access to that data only if the system is compatible with a security policy certified by a security administrator.

2.2 Trusted Channels

The standard approach for establishing secure channels over the Internet is to use security protocols such as Transport Layer Security (TLS) or Internet Protocol Security (IPSec), which aim at assuring confidentiality, integrity, and freshness of the transmitted data as well as authenticity of the endpoints involved. However, as mentioned before, secure channels do not provide any guarantees about the integrity of the communication endpoints, which can be compromised by viruses or Trojans. Based on security architectures that deploy Trusted Computing functionality, one can extend these protocols with integrity reporting mechanisms (e.g., the TLS extension proposed in [2]). Such extensions can be based on binary attestation or on property-based attestation.

2.3 Secure Network Virtualisation

Previous work on virtualising physical networks can be roughly grouped into two categories: those based on Ethernet virtualisation and those based on TCP/IP-level virtualisation. Although both categories include a substantial amount of work, few of these studies have an explicit focus on security. A secure network virtualisation framework was proposed by Cabuk *et al.* [6] for realising the network flow aspects of TVDs. The focus of [6] is a security-enhanced network virtualisation, which (1) allows groups of related VMs running on separate physical machines to be connected together as though they were on their own separate network fabric, and (2) enforces intra-TVD and inter-TVD security requirements such as confidentiality, integrity, and inter-TVD flow control. This has been achieved by an automatic provisioning of networking components such as VPNs, Ethernet encapsulation, VLAN tagging, and virtual firewalls. A second concept for managing VLAN access has been proposed by Berger *et al.* in [4]. Both papers contain similar concepts for managing VLANs inside the datacentre with some differences. The work of Berger *et al.* has more of a focus on integrity assurance using Trusted Computing. The work of Cabuk *et al.* [6] allows provisioning of secure virtual networking even if no VLAN infrastructure is present.

3 Unified policy enforcement for virtual datacentres

This section introduces a TVD-based policy enforcement framework that orchestrates the deployment and enforcement of the type of policies suitable for use across the datacentre. Policies alone are not sufficient to enforce customer separation in a virtualised datacenter. Ultimately, one needs to transform these policies into datacenter configurations and security mechanisms specific to each resource (e.g., VLAN configuration). To do so, the paper introduces a policy management scheme that accepts high-level domain policies and transforms them into resource-specific low-level policies and configurations.

The high-level policy defines the basic flow control, protection, and admission requirements. It aims at enforcing these high-level objectives throughout all resources in the datacentre. In the high-level model, flow control across customer domains can be specified by a simple matrix that defines whether flows are permitted. This however is not sufficiently fine-grained for specific resources. TVDs, for example, want to restrict their flow across boundaries by means of firewall rules. As a consequence, it is necessary to introduce a notion of policy refinement [24], because as translation moves towards lower levels of abstraction, it will require additional information (e.g., physical arrangement of the datacenter, "subjective" trust information) to be correctly and coherently executed. Proposed notion of policy refinement mandates the enforcement of "no flow" objectives while allowing each resource to refine what it means so that flows are permitted and how exactly unauthorised flows shall be prevented. Similarly, it does not allow resources to deviate from the confidentiality/integrity objectives; however, certain resources can be declared trusted so that they may enforce these objectives without additional security mechanisms such as encryption or authentication.

3.1 Network Security Policies

This section details the policy model of [6] and show how it related to the corresponding high-level policy. Similar to our high-level policies, there are two types of policies governing security in the network. The first limits flow between networks, whereas the second defines membership requirements to each network.

Network Security Policies across TVDs: A policy covers isolation and flow control between TVDs as well as integrity and confidentiality against outsiders. These basic security requirements are then mapped to appropriate policies for each resource. For example, from a networking perspective, isolation refers to the requirement that, unless the inter-TVD policies explicitly allow such an information flow, a dishonest VM in one TVD cannot (1) send messages to a dishonest VM in another TVD (information flow), (2) read messages sent on another TVD (confidentiality), (3) alter messages transmitted on another TVD (data integrity), and (4) become a member of another TVD network (access control). An information flow control matrix is a simple way of formalising these network connectivity objectives.

Intra-TVD Network Security Policy: Within a TVD, all VMs can freely communicate with each other while observing TVD-specific integrity and confidentiality requirements. For this purpose, the underlying infrastructure may ensure that intra-TVD communication only takes place over an authenticated and encrypted channel (e.g., IPSec), or alternatively, a trusted network.

3.2 Towards Storage Security Policies

Virtual disks attached to VMs must retain the advantages offered by storage virtualisation while at the same time enforcing TVD security policies. Advantages of storage virtualisation include improved storage utilisation, simplified storage administration, and the flexibility to accommodate heterogeneous physical storage devices. Similar to network, the following show a refinement of the high-level TVD policies into access control policies for VMs in certain roles to disks belonging to a domain.

Inter-TVD Storage Security: A virtual disk has a single label corresponding to the TVD it belongs to. Whenever a virtual machine operates on virtual storage, the global flow matrix described needs to be satisfied. For flexibility, each TVD can define a set of storage policies that govern usage and security of its storage. A single policy is then assigned to and enforced for each storage volume. Note that as flow within a domain is always allowed, this implies that disks of the same domain as the machine may always be mounted read/write.

Intra-TVD Storage Security: By default, this work considers the content of a disk to be confidential while the storage medium (possibly remote) is deemed to be untrusted. As a consequence, if a given domain does not declare a given storage medium as trusted, it deploys whole-disk encryption using a key that is maintained by the TVD infrastructure. Another aspect reflected in the disk policies is the fact that a possible notion of blank disks. Once they are written by another domain, they change color, and are then associated with this other domain while being encrypted under the corresponding key. In the future, it would be desirable to have integrity-protected storage where the TVD can validate that its content have not been changed by untrusted entities. For protecting the data in a particular TVD, virtual storage may in addition specify which conditions on the system must be satisfied before a disk may be *re-mounted* by a VM that has previously unmounted the disk, and whether shared mounting by multiple systems is allowed. Note that these membership restrictions require bookkeeping of disks and management of access of VMs to disks.

Therefore the existing storage and network virtualisation technologies as well as existing Trusted Computing components (in software and hardware) are the building blocks of the provided solution.

This framework (1) combines these technologies to realise TVDs and (2) orchestrates them using the TVD infrastructure, which provisions the appropriate security mechanisms.

3.3 TVD Infrastructure

The TVD infrastructure consists of a management layer and an enforcement layer. The TVD management layer includes TVD masters, proxies, and factories, whereas the TVD enforcement layer consists of various security services (figure 3.1). Each TVD is identified by a unique *TVD Master* that orchestrates TVD deployment and configuration. The TVD Master can be implemented as a centralised entity or have a distributed fault-tolerant implementation. The TVD Master contains a repository of high-level TVD policies and credentials (e.g., VPN keys). The Master also exposes a TVD management API through which the TVD owner can specify those policies and credentials. In the deployment phase, the TVD Master first verifies the suitability and capability of the physical host (which we refer to as pre-admission control). It then uses a generic *TVD Factory* service to spawn a *TVD Proxy*, which acts as the local delegate of the TVD Master dedicated to that particular host. The TVD Proxy is responsible for (1) translation of high-level TVD policies into low-level platform-specific configurations, (2) configuration of the host and security services with respect to the translated policies, and (3) interaction with the security services in TVD admission and flow control.

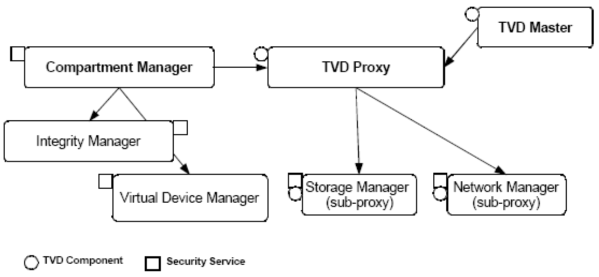


Figure 3.1: TVD Components and Security Services [7].

Security services implement the security enforcement layer of our TVD infrastructure. They run in a trusted execution environment on each physical host (e.g., Domain-0 in Xen) and (1) manage the security configuration of the hypervisor, (2) provide secure virtualisation of resources (e.g., virtual devices) to the VMs, and (3) provide support to TVD proxies in enforcing flow and access control policies within and across TVD boundaries. Figure 3.1 shows a high-level list of security services and their interaction with the TVD components. Most importantly, the *compartment manager* service manages the lifecycle of VMs in both para-virtualised and fully virtualised modes. This service works in collaboration with the TVD Proxy to admit VMs into TVDs. The *integrity manager* service implements Trusted Computing extensions and assists the TVD Proxy in host pre-admission and VM admission control. The *virtual network manager* and *virtual storage manager* services are invoked by the TVD Proxy. They implement resource virtualisation technologies and enforce parts of the high-level TVD policies that are relevant to their operation. Lastly, the *virtual device manager* service handles the secure resource allocation and setup of virtual devices assigned to each VM. This TVD infrastructure is geared towards automated deployment and enforcement of security policies specified by the TVD Master. Automated refinement and translation of high-level policies into low-level configurations are of particular interest. For example, for information flow between two hosts in a trusted datacentre environment, other mechanisms need to be in place than for a flow between two hosts at opposite ends of an untrusted WAN link. In the latter case, the hosts should be configured to allow communication between them only through a VPN tunnel.

Another important consideration is policy conflict detection and resolution [24]. In fact, conflicting high-level policies (e.g., a connection being allowed in the inter-TVD policy but disallowed in the intra-TVD policy) can potentially result in an incorrect configuration of the underlying infrastructure. It is not possible to solely rely on the TVD owner to specify conflict-free policies. It is important to detect policy conflicts and provide feedback to the owner in case one is detected. In the present prototype, policy refinement is performed manually. The result is a set of configuration files that are used for configuring the security services at the policy enforcement layer (e.g., the virtual networking infrastructure). In future work, it is adamant to investigate the automation of this step using, for example, the IETF policy model and various graph-based mechanisms from the literature. Furthermore, it is important to investigate different techniques for resolving conflicting policies [9].

3.4 Virtual Networking Infrastructure

Virtual networking (VNET) technologies enable the seamless interconnection of VMs that reside on different physical hosts as if they were running on the same machine. The proposed TVD framework, employs multiple technologies, including virtual switches, Ethernet encapsulation, VLAN tagging, and VPNs, to virtualise the underlying network and securely group VMs that belong to the same TVD. A single private virtual network is dedicated to each TVD, and network separation is ensured by connecting the VMs at the Ethernet level. Logically speaking, it provides a separate "virtual infrastructure" for each TVD in which it is possible to control and limit the sharing of network resources (such as routers, switches) between TVDs. This also provides the TVD owner with the freedom to deploy a wide range of networking solutions on top of the TVD network infrastructure. Network address allocations, transport protocols, and other services are then fully customisable by the TVD owner and work transparently as if the VMs were in an isolated physical network. To maintain secrecy and confidentiality of network data (where necessary), network communication is established over encrypted VPN tunnels. This enables the transparent use of untrusted networks between physical hosts that contain VMs within the same TVD to provide a seamless view of the TVD network. The following subsections introduce the technologies that have been used to implement a security enhanced VNET infrastructure for TVD owners. The concept of virtual switching is central to our architecture, which is then protected by existing VPN technologies that provide data confidentiality and integrity where needed. The VNET infrastructure acts as the local enforcer of VNET policies. As described in Section 3.1, these policies are based on the high-level TVD policies and translated into network configurations by the TVD Proxy. The Proxy then deploys the whole VNET infrastructure with respect to the translated configuration.

3.4.1 Virtual Switching

The *virtual switch* (vSwitch) is the central component of the virtual networking infrastructure and operates similarly to a physical switch. It is responsible for network virtualisation and isolation, and enables a virtual network to span multiple physical hosts. To do so, the vSwitch uses EtherIP and VLAN tagging to insert VLAN membership information into every network packet. The vSwitch also implements the necessary address-mapping techniques to direct packets only to those machines that host member VMs. Virtual switches provide the primitives for implementing higher level security policies for networking and are configured by the higher-level TVD management layer.

Figure 3.2 illustrates an example architecture in which physical machines host multiple VMs with different TVD memberships (the light and dark shades indicate different TVDs). Hosts A, B, and D are virtualised machines, whereas Host C is non-virtualised. Furthermore, Hosts A, B, and C reside on the same LAN, and thus can communicate directly using the trusted physical infrastructure without further protection (e.g., traffic encryption). For example, the *light* VMs hosted on Hosts A and B are inter-connected using the local VLAN-enabled physical switch. In this case, the physical switch separates the TVD traffic from other traffic passing through the switch using VLAN tags. Similarly, the *dark* VMs hosted on Host A and the non-virtualised Host C are seamlessly inter-connected using the local switch. In contrast, connections that require IP connectivity are routed over the WAN link.

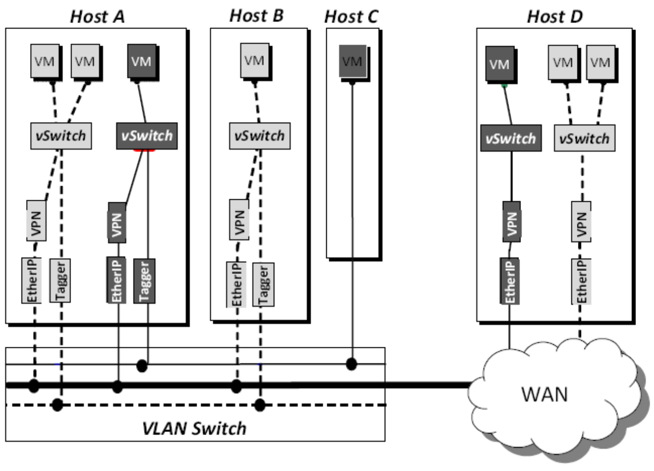


Figure 3.2: General vSwitch Architecture [7].

The WAN cloud in Figure 3.2 represents the physical network infrastructure able to deal with TVD-enabled virtual networks; it can include LANs with devices capable of VLAN tagging and gateways to connect the LANs to each other over (possibly insecure) WAN links. For connections that traverse untrusted medium, this research employs EtherIP encapsulation to denote TVD membership and additional security measures (such as encryption) to ensure compliance with the confidentiality and integrity requirements.

3.4.2 Virtual Private Networking

In Figure 3.2, VMs hosted on Host D are connected to the other machines over a WAN link. A practical setting in which such a connection might exist would be an outsourced remote resource connected to the local data centre through the Internet. As an example, lightly shaded VMs on Host D connect to the lone VM on Host B over this untrusted link. In a similar setting, this research uses a combination of EtherIP encapsulation and VPN technology to ensure the confidentiality and integrity of the communication. To do so, it uses point-to-point VPN tunnels with OpenVPN that are configured via the TVD Proxy from the TVD policies. This enables reconfiguration of the topology and the involved VPNs within a TVD from a single administration point, the TVD Master. TVD policies distributed from the TVD Master to the TVD Proxy also include the secret key for the VPN along with other VPN-specific settings. On a physical host, the VPN's endpoint is represented as a local virtual network interface (vif) that is plugged into the appropriate vSwitch controlled by the TVD Proxy. The vSwitch then decides whether to tunnel the communication between VMs, and if so, uses the VPN module to establish the tunnel and access the VPN secret for traffic encryption and decryption.

3.5 Virtual Storage Infrastructure

This research focuses on a simplified security management of virtualised storage. Broadly speaking, storage virtualisation abstracts away the physical storage resource(s). It is desirable to allow a storage resource to be shared by multiple host computers, and also to provide a single storage device abstraction to a computer irrespective of the underlying physical storage, which may be a single hard disk, a set of hard disks, a Storage Area Network (SAN), etc. To satisfy both requirements, storage virtualisation is typically done at two levels. The first level of virtualisation involves aggregating all the (potentially heterogeneous) physical storage devices into one or more virtual storage pools. The aggregation allows more centralised and convenient data management. The second level of virtualisation concerns the unified granularity (i.e., blocks or files) at which data in each pool is presented to the higher-level entities (operating systems, applications, or VMs).

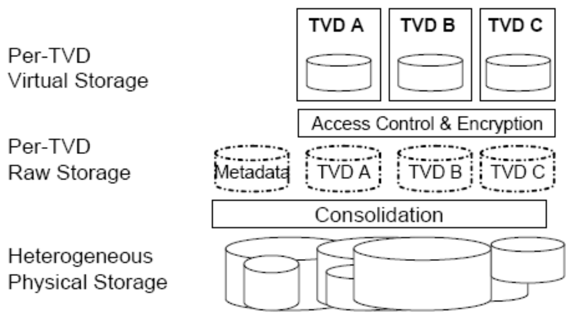


Figure 3.3: Security Enforcement for Virtualised Storage [7].

Figure 3.3 shows the proposed storage security enforcement architecture, in which existing heterogeneous physical storage devices are consolidated into a joint pool. This virtual storage pool is then subdivided into raw storage for each TVD. Each raw storage volume has an owner TVD that determines its policy (indicated by the labels TVD A, TVD B, and TVD C at the per-TVD raw storage layer in the figure). In addition, when a volume shall be shared among multiple TVDs, there is also a set of member TVDs associated with it. The access control and encryption layer helps enforce the storage-sharing policy defined by the owner TVD, e.g., enforcing read, write, create, and update access permissions for the member TVDs. This layer is a logical layer that in reality consists of the virtual storage managers (part of the security services) located on each physical platform. The virtual storage manager on each physical platform is responsible for enforcing the owner TVD's storage security policies (see Section 3.2) on these volumes. If a certain intra-TVD security policy requires confidentiality and does not declare the medium as trusted, the disk is encrypted using a key belonging to the owner TVD. If conditions for (re-)mounting a disk have been defined, the disk is also encrypted and the key is sealed against the TCB while including these conditions into the unsealing instructions. The policy and meta-data are held on a separate raw volume that is only accessible by the data centre infrastructure.

An administrator of a domain may request that a disk be mounted to a particular VM in a particular mode (read/write). As an example, in Xen, the disk is usually mounted in the management machine Domain-0 as a *back-end device* and then accessed by a guest VM via a *front-end* device. The virtual storage manager on the platform validates the mount request against the policies of both the TVD the VM is part of and the owner TVD for the disk. Once mounted, appropriate read-write permissions are granted based on the flow control policy for the two TVDs, e.g., read access is granted only if the policies specified in the disk policy matrix allow the VM's TVD such an access to the disk belonging to the owner TVD.

3.6 TVD Admission Control

When a VM is about to join a TVD, different properties will be verified by the local TVD Proxy to ensure that policies of all the TVDs that the VM is currently a member of as well as of the TVD that it wants to join are not violated. If the verification is successful, then the VM will be connected to that TVD. The TVD admission control protocol is the procedure by which the VM gets connected to the TVD. In the case of a VM joining multiple TVDs, the admission control protocol is executed for each of those TVDs. The following describes the steps of this protocol. It is assumed that the computing platform that executes the VM provides mechanisms that allow remote parties to convince themselves about its trustworthiness. Example mechanisms include trusted (authenticated) boot and the remote attestation protocol (see Section 2.1) based on TPM technology.

TVD Proxy Initialisation Phase: To allow a VM to join a TVD, the platform hosting the VM needs access to the TVD policy, and upon successful admission, to TVD secrets, such as the VPN key. For this purpose, TVD Proxy services are started on the platform for each TVD whose VMs may be hosted. The TVD Proxy can be started at boot time of the underlying hypervisor, by a system service (TVD Proxy Factory), or by the VM itself, as long as the TVD Proxy is strongly isolated from the VM.

Pre-Admission Phase: When a VM wants to join a TVD that is going to be hosted on the platform for the first time, the TVD Master has to establish a trust relationship with the platform running the VM, specifically with the TVD Proxy. This step is defined as the *pre-admission* phase. It involves the establishment of a trusted channel (see Section 2.2) between the TVDMaster and the TVD Proxy (or the TVD Proxy Factory). The trusted channel allows the TVD Master to verify the integrity of the TVD Proxy (Factory) and the underlying platform. After the trusted channel has been established and the correct configuration of the Proxy has been verified, the TVD Master can send the TVD policies and credentials (such as a VPN key) to the TVD Proxy.

Admission Control Phase: The Compartment Manager (part of the platform security services shown in Figure 3.1) is responsible for starting new VMs. The Compartment Manager loads the VM configuration and enforces the security directives with the help of the Integrity Manager (see Figure 3.1). The security directives may include gathering the VM state information, such as the VM configuration, kernel, and disk(s) that are going to be attached to the VM. If the VM configuration states that the VM should join one or more TVDs, then the Compartment Manager interacts with the corresponding TVD Proxy(ies) and invokes TPM functions to attest the state of the VM. The TVD Proxy verifies certain properties before allowing the VM to join the TVD. More concretely, the TVD Proxy has to ensure that;

- the VM fulfils the integrity requirements of the TVD;
- the information flow policies of all TVDs the VM will be a member of will not be violated;
- the VM enforces specific information flow rules between TVDs if such rules are required by the TVD policy, and that;
- the underlying platform (e.g., the hypervisor and attached devices) fulfils the security requirements of the TVD.

Platform verification involves matching the security requirements with the platform's capabilities and mechanisms instantiated on top of these capabilities. For example, suppose that data confidentiality is a

TVD requirement. Then, if hard disks or network connections are not trusted, additional mechanisms, such as block encryption or VPN (respectively), need to be instantiated to satisfy the requirement.

TVD Join Phase: If the VM and the provided infrastructure fulfil all TVD requirements, a new network stack is created and configured as described in Section 3.4. Once the Compartment Manager has started the VM, it sends an attach request to the corresponding TVD vSwitch. Once the VM is connected to the vSwitch, it is a member of the TVD.

4. Conclusions

This paper describes the components for a Trusted Virtual Datacentre. The core idea is to build a datacentre that allows for and isolates multiple customers while satisfying the security requirements of each customer. Securing the access to data on persistent media and during transfer over the network is a serious problem in distributed virtual datacentre and cloud computing scenarios. The research proposes a framework based on TVDs and Trusted Computing for secure network and storage virtualisation that includes mechanisms for verifying the integrity of the hypervisor, VMs, and security policy enforcement points. This is achieved by means of so-called Trusted Virtual Domains that provide a virtual environment for each customer. Each trusted virtual domain exposes a standardised management interface that allows customers to run existing management software to manage their respective domains. The concept of TVDs is rigid enough to allow consistent policy enforcement across a group of domain elements, while being flexible enough to support policy-controlled interactions between different TVDs. TVD policies and configurations are ‘backward-compatible’ in supporting options that could be taken for granted in non-virtualised data centres. For example, co-hosting of specific customer services with those of other datacentre customers on the same physical platform could be inhibited if so desired. By incorporating hardware-based Trusted Computing technology, this framework allows the creation of policy domains with attestable trust properties for each of the domain nodes. In order to enforce the given security requirements, this research deploys the security services described in [20] that allow verifiable security within each domain. Further it includes validation of the trusted computing base as well as verifiable enforcement of given per-domain policies. The inclusion of integrity measurement and management mechanisms as part of the physical platform’s TCBs provides both datacentre customers and administrators with a much needed view of the elements (hypervisors, VMs, etc.) that are part of their virtual infrastructure as well as information on the configurations of those elements. The proposed framework can be used to obtain information about the elements on a ‘need-to-know’ basis without having to introduce all-powerful roles of administrators with access to every aspect of a platform. This work have substantially contributed to the knowledge in secure virtualisation and trusted computing. Overall, the researchers managed to build the first large-scale virtual datacentre that can be validated using trusted computing technology. Nevertheless, despite the substantial progress, there are still several open research challenges. Examples include large-scale management and validation of integrity as well as efficient implementation strong isolation guarantees on a finer-grained level. In the long run, researchers hope that this will enable to lower the granularity of integrity guarantees from Virtual Machines to individual objects.

Acknowledgement: This article is constructed based on the following documentation in order to promote and discuss the techniques presented amongst the members and participants of UKUUG 2010.

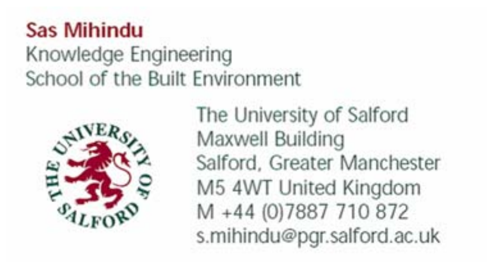
(1) OpenTC Deliverable D05.6 (2009) OpenTC Workpackage 5, Final R7628/2009/01/15/OpenTC European Commission Project: IST-027635

(2) Serdar Cabuk et al (2010) Towards automated security policy enforcement (Eds) Jan Camenisch, Javier Lopez, Fabio Massacci, Massimo Ciscato and Thomas Skordas, JCS special issue on EU-funded ICT research on Trust and Security, Journal of Computer Security 18 (2010) 89–121

(The above paper (2) is one of the six articles presented in a special issue of the *Journal of Computer Security* (IOS Press, ISSN 0926-227X, http://iospress.metapress.com/content/x1766147m103/) brings together the research results of six, FP6-IST projects.)

Author/Editor:

The author has developed skills and experience by working for academic, research and business enterprises in Europe, and Australasia over two decades and is the founding member of UK's first 'BIM Research & Special Interest Group'. His current research interests are human-centric technology infrastructures for future workplaces (FWS), strategies of distributed knowledge and data management for community collaboration and virtual reality, virtual environments and virtual technology infrastructures. He has been leading and delivering for many European Commission projects with Information Society Technologies (IST) Programmes via University of Salford, Greater Manchester. He has recently been engaged in the design and deployment of Internet infrastructures for technological observatory of VR, VE and FWS, and Strategic and FWS, and Strategic Knowledge Management infrastructures for scientific and professional communities. In addition to this he has also written many research papers on Knowledge Engineering, Communication and Collaboration Tools and Distributed Virtual Networking infrastructures for collaboration. One of his current research goals is to establish a framework for Virtual Technology Infrastructures to support effective communication and collaboration for virtual communities. He can be contacted through the details below or via his personal e-box at; mybox@sas.mihindu.name.



Sas Mihindu
Knowledge Engineering
School of the Built Environment
The University of Salford
Maxwell Building
Salford, Greater Manchester
M5 4WT United Kingdom
M +44 (0)7887 710 872
s.mihindu@pgr.salford.ac.uk

Bibliography

[1] M. J. Anderson,M. Moffie, and C. I. Dalton. Towards trustworthy virtualization environments: Xen library os security service infrastructure. Research report, HP Labs, Bristol, UK, 2007.
[2] F. Armknecht, Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, G. Ramunno, and D. Vernizzi. An efficient implementation of Trusted Channels based on Openssl. In STC ’08: Proceedings of the 3rd ACMworkshop on Scalable Trusted Computing, pages 41–50, New York, NY, USA, 2008. ACM Press.
[3] S. Berger, R. Cáceres, K. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the Trusted Platform Module. In Proc. 15th USENIX Security Symposium, pages 21–21, Aug. 2006.
[4] S. Berger, R. Cáceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan. TVDc: managing security in the trusted virtual datacenter. SIGOPS Operating Systems Review, 42(1):40–47, 2008.
[5] A. Bussani, J. L. Griffin, B. Jansen, K. Julisch, G. Karjoth, H. Maruyama, M. Nakamura, R. Perez, M. Schunter, A. Tanner, L. van Doorn, E. V. Herreweghen, M. Waidner, and S. Yoshihama. Trusted Virtual Domains: Secure foundation for business and IT services. Research Report RC 23792, IBM Research,
Nov. 2005.
[6] S. Cabuk, C. Dalton, H. V. Ramasamy, and M. Schunter. Towards automated provisioning of secure virtualized networks. In Proc. 14th ACM Conference on Computer and Communications Security (CCS-2007), pages 235–245, Oct. 2007.
[7] S. Cabuk, et al (2010) Towards automated security policy enforcement (Eds) Jan Camenisch, Javier Lopez, Fabio Massacci, Massimo Ciscato and Thomas Skordas, JCS special issue on EU-funded ICT research on Trust and Security, Journal of Computer Security 18 (2010) 89–121
[8] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In Proc. 2nd Symposium on Networked Systems Design and Implementation (NSDI-2005), pages 273–286, May 2005.
[9] N. Dunlop, J. Indulska, and K. A. Raymond. A formal specification of conflicts in dynamic policy-based management systems. DSTC Technical Report, CRC for Enterprise Distributed Systems, University of Queensland, Australia, Aug. 2001.
[10] P. England, B. Lampson, J. Manferdelli, and B. Willman. A trusted open platform. Computer, 36(7):55–62, 2003.
[11] European Multilaterally Secure Computing Base (EMSCB) Project. Towards Trustworthy Systems with Open Standards and Trusted Computing, 2008. http://www.emscb.de.
[12] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In ACM Symposium on Operating Systems Principles (ASOSP), pages 193–206. ACM Press, 2003.
[13] J. Griffin, T. Jaeger, R. Perez, R. Sailer, L. V. Doorn, and R. Caceres. Trusted Virtual Domains: Toward secure distributed services. In Proc. 1st Workshop on Hot Topics in System Dependability (Hotdep-2005), Yokohama, Japan, June 2005. IEEE Press.
[14] V. Haldar, D. Chandra, and M. Franz. Semantic Remote Attestation – virtual machine directed approach to Trusted Computing. In USENIX Virtual Machine Research and Technology Symposium, pages 29–41, 2004. also Technical Report No. 03-20, School of Information and Computer Science, University of
California, Irvine.
[15] Y. Katsuno, M. Kudo, Y. Watanabe, S. Yoshihama, R. Perez, R. Sailer, and L. van Doorn. Towards Multi–Layer Trusted Virtual Domains. In The Second Workshop on Advances in Trusted Computing (WATC ’06 Fall), Tokyo, Japan,
[16] D. Kuhlmann, R. Landfermann, H. Ramasamy, M. Schunter, G. Ramunno, and D. Vernizzi. An Open Trusted Computing Architecture - Secure virtual machines enabling user-defined policy enforcement,
2006. http://www.opentc.net/images/otc_architecture_
high_level_overview.pdf.
[17] U. Kühn, M. Selhorst, and C. Stüble. Realizing property-based attestation and sealing with commonly available hard- and software. In STC ’07: Proceedings of the 2007 ACM workshop on Scalable trusted computing, pages 50–57, New York, NY, USA, 2007. ACM.
[18] Open Trusted Computing (OpenTC) Project. The OpenTC Project Homepage,
2008. http://www.opentc.net/.
[19] OpenSSL Project. The OpenSSL Project Homepage, 2007. http://www.openssl.org/.
[20] OpenTCWorkpackage 05. Design of the cross-domain security services. Deliverable D05.4, The OpenTC Project www.opentc.net, 05/26/2008.
[21] A.-R. Sadeghi and C. Stüble. Property-based Attestation for Computing Platforms: Caring about Properties, not Mechanisms. In Proc. 2004 Workshop on New Security Paradigms (NSPW-2004), pages 67–77, New York, NY, USA, 2005. ACM Press.
[22] Trusted Computing Group (TCG). www.trustedcomputinggroup.org.
[23] Trusted Computing Group (TCG). TCG TPM specification version 1.2 revision 103.
https://www.trustedcomputinggroup.org/specs/
TPM/, July 2007. See also [122] and http://www.trustedcomputing.org/.
[24] A.Westerinen (2001) Terminology for policy-based management. RFC 3198, November 2001.