Private Virtual Infrastructure: A Model for Trustworthy Utility Cloud Computing

UMBC Computer Science Technical Report Number TR-CS-10-04

F. John Krautheim[1] Dhananjay S. Phatak Alan T. Sherman[1]

Cyber Defense Lab, Dept. of CSEE
University of Maryland, Baltimore County (UMBC)
Baltimore, MD 21250

john.krautheim@umbc.edu phatak@umbc.edu sherman@umbc.edu

[1] Krautheim and Sherman were supported in part by the Department of Defense under Information Assurance Scholarship Program grants H98230-08-1-0334 and H98230-09-1-0404.

Abstract

Private Virtual Infrastructure is a security architecture for cloud computing which uses a new trust model to share the responsibility of security in cloud computing between the service provider and client, decreasing the risk exposure to both. Private Virtual Infrastructure is under control of the information owner while the cloud fabric is under control of the service provider. The Private Virtual Infrastructure architecture comprises a cluster of trusted computing fabric platforms that host virtual servers running an application server with a Locator Bot security service. The cloud Locator Bot pre-measures the cloud platform for security properties to determine the trustworthiness of the platform. The Locator Bot uses Trusted Execution Technology and virtual Trusted Platform Modules to pre-measure the target environment and securely provision the Private Virtual Infrastructure in the cloud thus protecting information by preventing data from being placed in malicious or untrusted environments. Private Virtual Infrastructure — including Locator Bot — provides organizations tools to maintain control of their information in the cloud and realize benefits of cloud computing, with assurance that their information is protected. This paper presents a cloud trust model, Private Virtual Infrastructure architecture, and a Locator Bot protocol in enough detail to support further analysis or implementation.

Keywords: Cloud Computing; Security; Architecture; Virtualization, Trusted Computing, Trusted Platform Module; Networking

1 Introduction

Cloud computing requires a new trust paradigm that shares responsibility for security between providers and consumers of cloud computing services. *Private Virtual Infrastructure* (PVI) first introduced in (Krautheim, 2009) is our approach to managing trust and security in cloud computing environments. We introduce *Locator Bot* (LoBot) to implement the PVI on cloud resources with a level of assurance that is required to meet data confidentiality and privacy concerns of sensitive information. LoBot is a virtual machine (VM) appliance that acts as an agent to locate trustworthy platforms, provides a root of trust for PVI *virtual servers* (VSs) via a *Virtual Trusted Platform Module* (VTPM), and provisions PVI within the cloud.

Cloud computing is poised to revolutionize computing as a service where *Information Technology* (IT) becomes a computing utility (Carr, 2008) delivered over the Internet. Utilizing massively scalable computing resources delivered as a service using Internet technologies, cloud computing provides the ability to deliver on-demand computing resources dynamically. Companies can fundamentally change their information technology strategy as they turn to the cloud for datacenter and information technology services to improve scalability and global reach, and to lower overhead. There are many benefits to cloud computing including lower overall cost of IT ownership, increased flexibility, fault tolerance, locality flexibility ability, and to respond to new business requirements quickly and efficiently. As with any new technology, this new way of operating brings with it new risks and challenges, especially when considering the security and privacy of information stored and processed within the cloud. Utility cloud computing resources are shared among a large number of consumers to allow for a lower cost of ownership; however, current technologies lack adequate security, potentially exposing information stored in the cloud to all the other users of the cloud. PVI is our solution for managing trust and security that allows organizations with data confidentiality concerns to leverage the economies of scale of cloud computing technologies.

Many organizations such as financial institutions, health care providers, and government agencies are legally required to protect their data from compromise due to its sensitivity. Consider a healthcare provider that wants to move a database of patient medical records to a utility cloud to allow patients and insurance companies easier access to the information. Medical records are considered private and must be protected under the Health Insurance Portability and Accountability Act (HIPAA) of 1996; therefore, medical records in the cloud must be protected from unauthorized disclosure. A utility cloud potentially exposes the information to the operators of the cloud, other users of the cloud, and anyone who has access to the Internet. Typically, organizations with privacy requirements manage and maintain their own datacenters with stringent physical and logical protection mechanisms ensuring that their data remains protected. These organizations simply cannot use cloud computing in a generic manner due to the inherent risk of data compromise from systems they do not control. PVI provides these organizations a means to manage security in a utility cloud by facilitating sharing of security management roles and responsibilities between service provider and customer. PVI for cloud computing enables organizations to maintain control of their information in the cloud and realize benefits of cloud computing.

Private Virtual Infrastructure is an architecture for utility cloud computing that meets the information owner's requirements for data protection and privacy. The PVI architecture is a cluster of *trusted computing fabric platforms* (TCFPs) that are owned, operated and configured by cloud service providers. The TCFPs host VSs for clients, which are two tightly coupled virtual machines, an *application server* (AS), and a LoBot that provides security services for the AS. Each PVI instance has a single trusted central control and policy decision point called the *PVI Factory* (PVIF), which provides the root of trust for PVI. The PVIF maintains and manages keys for PVI and serves as the certificate authority for the LoBots' *Endorsement Keys* (EKs).

LoBot answers the question of whether a cloud computing platform is trustworthy. While determining whether a particular cloud platform is secure is an undecidable problem, there are certain properties we can measure to help determine a level of trustworthiness of the platform. These properties include the EK and *Attestation Identity Keys* (AIK) of the cloud platform, an attestation of security properties from the TCFP, and an attestation of the configuration of the TCFP. Using these property measurements from the LoBot, the PVIF can decide whether to trust a particular cloud platform. Once the level of trustworthiness is determined, LoBot provides secure provisioning, migration, and monitoring services for the VSs in PVI.

The AS has trust rooted both in the TCFP and in the PVIF. The TCFP's physical *Trusted Platform Module* (TPM) binds the LoBot's VTPM to the TCFP's root of trust. The LoBot's VTPM trust is rooted with the PVIF that generates its EK and VTPM proof. This trust relationship is a new model that removes the root of trust for the AS solely from the physical server by adding a component from the virtual infrastructure. Typical trusted computing techniques have a single root of trust in the TPM, but due to the

Acronym	Meaning
AIK	Attestation Identity Key
AS	Application Server
CVF	Cloud Virtual Fabric
EK	Endorsement Key
IaaS	Infrastructure as a Service
LoBot	Locator Bot
LSP	LoBot Secure Provisioning
MLE	Measure Launch Environment
PCR	Platform Configuration Register
PVI	Private Virtual Infrastructure
SAN	Storage Area Network
SRK	Storage Root Key
TCB	Trusted Computing Base
TCFP	Trusted Cloud Fabric Platform
TCP	Trusted Computing Platform
TPM	Trusted Platform Module
TXT	Trusted eXecution Technology
VM	Virtual Machine
VS	Virtual Server
VTPM	Virtual Trusted Platform Module

Table 1 - Acronyms

unique nature of the cloud computing environment, we propose this new model of combined trust as a method to verify the authenticity of a virtual server in a cloud environment.

Contributions: This paper makes the following novel contributions:

- An architecture for protecting privacy and confidentiality in a cloud datacenter that separates security relationships between the user and consumer of cloud services and combines trust from both for the virtual and physical environments.

- The concept of an agent or “bot” virtual appliance to protect provisioning and migrations of virtual machines in a cloud datacenter through fabric pre-measurement.

- Protocols to ensure secure provisioning and secure live migration of virtual machines in the cloud datacenter.

This work represents a new paradigm of information protection and security in cloud computing. The usefulness of PVI and LoBot extends beyond cloud computing to any environment where remote virtual machine provisioning and live migrations are required.

Outline: This paper is organized as follows. Section 2 overviews cloud computing and introduces an illustrative example. Section 3 reviews background of related security research in the virtualization and cloud computing. Section 4 provides a background on trusted computing technologies. Section 5 describes the PVI architecture in detail. Section 6 presents the LoBot protocols. Section 7 analyzes how the components and protocols of PVI and LoBot work together to provide security and privacy in the cloud. Section 8 discusses follow-on and future work. Section 9 concludes the paper. Table 1 provides acronyms used throughout the paper.

2 Our Cloud Computing Model

Consider a landlord of an apartment building and the tenants of each individual unit. Tenants expect the landlord to provide basic services such as facility maintenance, security for the complex, while the landlord expects the tenants maintain their personal property and personal security (keep their doors locked). As long as the tenants uphold all the requirements of their lease agreement, they expect the landlord will respect the privacy and not spy on them or interfere with their personal property. A trust relationship is established between the landlord and tenant that each will uphold their end of the agreement.

In the cloud computing realm, the service provider can be thought of as the landlord and the client can be thought of as the tenant. *Infrastructure as a Service* (IaaS) (Vaquero, et al., 2009) is a model of cloud computing where all of the facilities required for datacenter applications are available over the Internet, which clients purchase as an outsourced service. An IaaS datacenter is similar to the apartment building above in that it provides infrastructure hosting for multiple customers. A datacenter could have the capability to host thousands of VSs for clients who rent or lease the servers much as apartment tenant. By using cloud IaaS as a datacenter replacement, companies can manage traffic and loading agility. The clients can run web servers, applications, or databases on their VSs, dynamically increasing or decreasing their server capacity as needed. As long as they abide by the service agreement, the provider should not interfere with the operation of the customers' VSs, monitor the clients' communications, or view or modify the customers' data. Since security is not an integral part of IaaS, and since the management and ownership of the hosting platforms is removed from the information owner, ensuring the security and integrity of information in IaaS is a major unresolved issue.

Let us consider a hypothetical example of a healthcare provider called CloudHealth. CloudHealth maintains a database of patients' records, insurance companies, and billing. CloudHealth needs to reduce overhead. Reasoning that being an IT tenant would be cheaper than being an owner, they choose Fabricorp as their service provider. Fabricorp is a reputable Internet company that has been in business since the early days of the Internet and recently moved into cloud datacenter services. Fabricorp built several massive datacenters, each with several thousand servers; thousands of individual clients may be using its cloud datacenter services at any given time. CloudHealth will move all of its corporate servers and data assets into Fabricorp's cloud, maintaining only thin clients, laptops or nettops, and mobile computing devices on site for doctors and administrators to access the cloud datacenter. This scenario allows CloudHealth to leverage large amounts of computing power at Fabricorp to run analytics on healthcare data and perform billing operations when needed without having to maintain those computing resources locally, thus considerably reducing their IT costs. The cloud datacenter contains all the personal and private information about CloudHealth's patients, relationships with other companies, and financial information. To comply with HIPAA regulations, CloudHealth needs to have assurances that its data is protected from Fabricorp's other cloud users, Fabricorp's personnel at the cloud datacenter, and other Internet users who have access to Fabricorp's services. We will show how PVI can be used to help CloudHealth leverage the benefits of cloud computing.

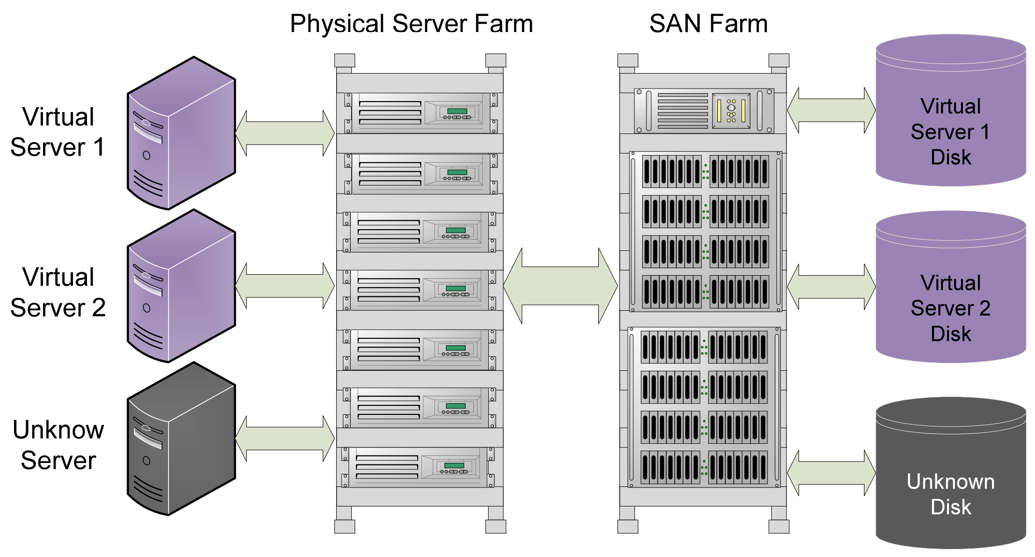

Figure 1 – Cloud Virtual Fabric topology consists of a server farm, SAN farm, virtual severs and virtual disk.

2.1 Cloud Virtual Fabric

Fabricorp uses an IaaS topology called the *Cloud Virtual Fabric* (CVF) for their cloud datacenter. CVF is a network of computing platforms that provides elastic cloud computing services (Nurmi, et al., 2008). Elastic cloud computing is defined as scalable on-demand computing resources delivered as a service over Internet technologies. An elastic cloud can have many different applications and servers, from many information owners, simultaneously running within the CVF. Each client's view of the cloud will be different from another client's view. No client should be able to see what any other client is doing within the cloud.

Figure 1 shows the topology of CVF which accommodates many VSs (known domains) and unknown domains (other users) on a fabric of computing platforms. These known and unknown domains need to be isolated from each other to provide privacy and security within the cloud. *Storage Area Networks* (SANs) provide virtual storage for the computing fabric and VSs. Virtual storage is accessed by VSs via a protocol like iSCSI, Fibre Channel over IP, or a similar protocol that physically separates storage from processing servers, providing the illusion of local storage for the VSs. This fabric configuration allows

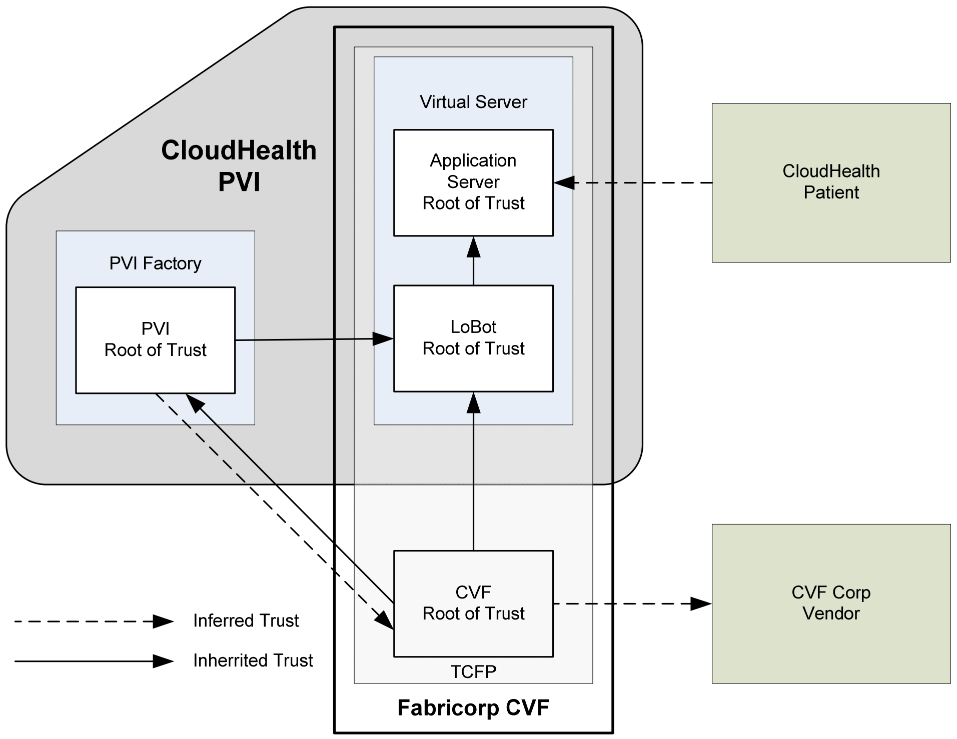

Figure 2 –Trust relationships in the cloud are complex with multiple relationships both inferred and inherited.

servers and storage to be spread across many physical facilities, while supporting dynamic provisioning and live migration capabilities for the VSs.

2.2 Cloud Trust Model

This cloud computing datacenter environment is laden with trust issues. The client must not only trust the datacenter operator, but each of individual clients that are collocated in the facility and each third party service provider of the operator. Clients needs assurance that their sensitive information is protected from compromise and loss and that their application is available when demanded. If only one of the thousands of client VSs in the datacenter were malicious, it would have to capability to compromise and threaten all the remaining clients in the datacenter destroying the trust and integrity of all parties involved.

We define a new model for trust in the cloud that accounts for various sources of trust and multiple trust relationships. Figure 2 show trust relationships that are present in our CloudHealth example. The cloud by definition is owned and operated by a second party, Fabricorp. A trust relationship is established that Fabricorp will provide trustworthy services to CloudHealth when they enter into a business relationship. This relationship is an *inferred trust* in that it is a social trust, not a logical trust. Other inferred trust relationships in the example are that patients trust CloudHealth with their medical records and Fabricorp trusts vendors to supply support services. CloudHealth's patients have no relationship with Fabricorp and want to be assured that their medical records are secure, which is CloudHealth's responsibility. A Fabricorp vendor, for example a billing processor, does not have a trust relationship with CloudHealth nor its patients, yet it is critical that the bills for Fabricorp's services are processed and the billing vendor does not interfere with CloudHealth's operations. Our trust model solves the problem of maintaining CloudHealth's trust with its ASs in FrabriCorp's CVF, and ultimately, its patients. This trust relationship must be maintained in the presence of parties who have no relationship with CloudHealth or its patients.

As can be seen from Figure 2, the trust relationships are not linear. An application server has two sources of trust, trust in the CVF (provider), and trust in the PVI (client). The PVIF first needs to establish a logical trust in the TCFP before it initiates any provisioning of PVI. This is done by inheriting the root of trust in the TCFP and combining it with the PVI root of trust with a LoBot. LoBot's compound trust relationship is then used as a root of trust for the VS. The VS's root of trust is distinct in that LoBot has combined two unique roots of trust into a third compound root of trust with attributes inherited from both sources. This LoBot root of trust is unique to the TCFP it is running on such that it cannot be cloned and used on another platform without unbinding the trust relationship to the TCFP. Additionally, the VS cannot be used by other entities in CVF, even on the same TCFP, as the trust is bound to PVI as well as the TCFP.

2.3 Cloud Adversarial Model

Potential adversaries in the cloud computing environment are cloud provider insiders, other tenants in the cloud, and outside attackers. The service provider insider acting as a cloud administrator has full privileges of the host machines and has the ability to attack a VM owned by a customer since she has higher system privilege than does the client. A cloud tenant can potentially eavesdrop on other tenants in the cloud or escalate his privilege by breaking containment and attacking other tenants. The outside attacker is generally the same as the outside attacker for a regular IT scenario with the exception that the attacker has a higher incentive to attack clouds since the opportunity for accessing valuable data is higher due to the concentration of users.

There are several attacks against the VMs that can be performed by malicious actors inside the CVF. A malicious administrator can surreptitiously attack a VM in the cloud using her higher privileged access to inspect memory, monitor VM communications, and perform suspend and reboot attacks. This can be very difficult to defend against, so we must be vigilant in determining the security settings and configuration of the host environment prior to provisioning.

An outside attacker could place a malicious VM in cloud. This malicious VM, or *malvm*, in the cloud can purport to be a valid host machine of the service provider and receive information environments from unsuspecting users. This malvm could also snoop on the traffic within the datacenter. Once an attacker is able to implant a malvm in CVF, the entire datacenter and potentially the entire cloud could be compromised. An example attack is a malvm spoofing itself as a valid host machine on the service provider's network. When a new server is requested in the cloud, the malvm receives the new VS from the customer, copies all confidential data, and transmits it back to the attacker. These actions are performed undetected without alerting the user that they are operating on a compromised machine. As far as the users know, their data assets are on a valid host server as it appears they sent the information to the service provider directly.

A similar attack we wish to defend against is the malicious host (*malhost*). In this scenario, an attacker gains control of a node by compromising the hypervisor or host OS. Once malicious control of the host is established, the malhost can accept new VS requests. The malhost is legitimate and on the provider's network; therefore, it is implicitly trusted and treated as a non-malicious host. Since the malicious hypervisor and host have higher privilege than the guest machine, the malicious host has access to all operations and memory of the guest and ability to compromise any sensitive information to which the guest has access.

A man-in-the-middle attack against VM provisioning or live migration is another attack we defend against. In this scenario, the attacker wishes to gain access to encrypted information on the storage array or database. The attacker listens for provisioning or migration requests by the target and intercepts the image and state information from the source machine. This interception can be accomplished through spoofing, ARP poisoning (Gibson, 2005), DNS poisoning (Gibson, Laporte, 2008), or any other attacks that redirect traffic from the destination machine to the attacker. Since the state information is a copy of the VM's memory and processor state, the attacker can manipulate the image and machine state by implanting malicious software and then placing the compromised image on destination server. The attacker now has access to the server's secure data stored on the SAN through the malicious software implant. We ensure that this attack is not possible by verifying the integrity of the image before and after provisioning.

2.4 PVI Security Model

In a cloud, traditional security methodologies do not work as the service providers cannot allow information owners, or clients, to manipulate the security settings of the fabric. If this were allowed, it would be possible for one client to change security settings illicitly in their favor, or change security settings of other clients maliciously. This situation is unacceptable since the information owner cannot manage the security posture of their computing environment. Therefore, a security model is needed that allows information owners to protect their data while not interfering with the privacy of other information owners within the cloud.

We propose a model for security that is shared between operators and clients. Operators need to give clients visibility into the security posture of the fabric while maintaining control of the fabric. The clients need to have assurance that they can control the privacy and confidentiality of their information at all times and have assurances that if needed, they can remove, destroy, or lock down their data at any time.

The PVI security model is a virtual datacenter over the existing cloud infrastructure. This virtual datacenter is under control of the information owner while the fabric is under control of the operator. Both parties must agree to share security information between themselves and possibly other parties in the cloud to achieve situational awareness of the security posture at all times.

The key concept of the PVI security model is the ability to verify security settings of the underlying fabric. The provider needs to provide security services which protect and monitor the fabric. These services can be reported to the virtual environment via a cryptographic protocol that attests these services. LoBot is the mechanism by which security information is measured, shared, and reported between the data owners and infrastructure operators.

Private Virtual Infrastructure meets the goals of a shared security posture where all resources necessary for the virtual datacenter are securely isolated from the greater cloud. LoBots provide secure provisioning of commodity internet resources within the PVI. Isolation of the client's virtual datacenter is accomplished through VM containment, encryption, and access control.

3 Related Work

This paper extends our previously published work on Private Virtual Infrastructure (Krautheim, 2009). New contributions include our analysis of trust in the cloud and more specific details on the design, architecture, and protocols required to implement PVI.

Virtualization is a fundamental technology for implementing IaaS cloud services and PVI. Virtualization is the capability to share a single physical computer among many simultaneous guest VMs via a virtual machine monitor, or *hypervisor*. Xen (Barham, et al., 2003) is an open source hypervisor that provides a the ability for VMs to run full operating systems as well as smaller "helper VMs" called stubs that provide services, such as hardware emulation for virtual TPMs.

There are many security issues that must be considered when using virtualization technology including separation of private data between VMs, virtualization containment attacks, and hypervisor attacks against the VM. IBM developed a secure hypervisor called sHype (Sailer, et al., 2005) to help solve some of these problems. SHype provides isolation between the VMs and controls resource sharing but does nothing to verify the integrity of the VMs launched on the host. Terra (Garfinkel, et al., 2003) is a trusted virtual machine monitor that can create isolated tamper resistant VMs on a host with the ability to verify the VMs; however, it does not provide the ability to measure the environment before provisioning of the virtual machine and does not guarantee a secure launch of the virtual environment. There are also disaggregation techniques developed to improve Xen security (Murray, et al., 2008) that reduce the trusted computing base of the hypervisor and VMs which is useful for our needs; but we still need to validate that the environment is properly configured prior to provisioning our VMs.

Going beyond virtual machine security, IBM's Integrity Measurement Architecture (IMA) (Sailer, et al., 2004) validates all executable content, from the hypervisor up to the application layer, by measuring the content before execution. IMA uses the TPM to store the measurements for attestation and validation at a later time. IMA provides guarantees that content integrity is maintained but provides no guarantees of confidentiality if the information owner decides not to trust the environment. IMA led to several proposals for endpoint integrity including (Gasmi, et al., 2007, Goldman, et al., 2006). These techniques measure the endpoint, set up secure tunnels, and verify security properties of the target to determine the trustworthiness of the endpoint. PVI via LoBot uses similar techniques to pre-measure a target environment by combining the cloud trust model with endpoint trustworthiness and provide secure provisioning services.

Distributed *Mandatory Access Control* (MAC) research by McCune (2006) defines a shared reference monitor that enforces MAC policies across a distributed set of machines, allowing the setting of a consistent security policies and access controls across them. PVI uses a form distributed MAC for the PVI layer, but MAC may or may not in place on the CVF layer; therefore, we must still determine the security properties of the host. Property based attestation (Sadeghi, Stüble, 2004) is one method to determine security properties of a host that allows a platform to attest to properties about itself rather than performing a binary measurement of the platform, thus providing more flexibility for attestation and more resiliency to changes to configuration setting and patches than measurement based attestation. A protocol for property based attestation developed by (Chen, et al., 2006) provides a means to perform remote property based attestations but does not provide any provisioning capabilities.

IBM's Trusted Virtual Datacenter (TVDc) (Berger, et al., 2008) incorporates trusted computing technologies into virtualization and systems management providing many features that can be used for securing a cloud datacenter including VM isolation via sHype, vTPM support, and system management software. TVDc uses Trusted Virtual Domains (Bussani, et al., 2005) to provide strong isolation and integrity guarantees that significantly enhance the security and management capabilities of virtualized environments. We extend the capabilities of TVDc with our PVI security model, architecture and LoBot provisioning protocol.

A proposal for trusted cloud computing made by Santos (2009) focuses on building trusted cloud computing platforms using the Trusted Computing Group's specifications. PVI does not focus on the host platform, rather the architecture for the virtual datacenter operating on the trusted cloud computing platforms. PVI also introduces our cloud trust model that accounts for trust beyond the trusted platform. Another work that closely resembles LoBot is an agent transfer protocol (Gallery, et al., 2009) for moving a trusted agent around the cloud between fixed and mobile platforms. LoBot is designed specifically for building trust in PVI and securing provisioning and live migration of virtual machines. LoBot has an entirely different purpose than the trusted agent, but there are many similarities in the way trust is established on the remote platform. Both agents use property based attestation to determine the security properties of the destination platform

4 Trusted Computing Capabilities

Trusted computing technology provides a robust foundation on which to build a secure cloud. A *trusted computing platform* (TCP) uses a trusted component to provide a foundation for trust for software processes (Pearson, 2003). The Trusted Computing Group's (TCG) specification (TCG, 2007) states that TCPs have a two roots of trust, a *Root of Trust for Measurement* (RTM) and a *Root of Trust for Reporting* (RTR). The RTM provides secure measurement of the platform and the RTR allows a verified report of the measurement through the process of *attestation*. On Intel platforms, the RTM is included in the BIOS boot block and the RTR is the TPM (Grawrock, 2009).

4.1 Trusted Platform Module

The TPM (TPM, 2007) is the cryptographic component of the TCP serving as the RTR. The TPM also serves as the *Root of Trust for Storage* (RTS) providing secure storage for the TCP to store encryption keys and other information. Other useful features of the TPM include a non-volatile random access memory (NVRAM) for secure storage of keys and user data, and a true random number generator for key and nonce generation.

Measurement of the platform is accomplished by performing a *Secure Hash Algorithm* (SHA-1) hash on the code loaded on the platform and storing the result in the TPM's *Platform Configuration Registers* (PCRs). The attestation process allows clients to request a quote of the PCRs signed by the TPM and verify that the platform they are using meets their policy and configuration requirements. Clients can then determine whether they wish to use the services provided by the platform based on the attestation from the platform's TPM.

Each TPM has an Endorsement Key and EK certificate that provides a unique identity for the TPM and validates that the TPM is legitimate. An Attestation Identity Key, signed by a privacy CA, can also be created that is used to verify that the TPM is legitimate; however, an AIK has no information that is uniquely identified to a single platform.

4.2 Virtual TPM

One problem associated with the TPM is that it only works for non-virtualized environments. Most TPM implementations closely follow the TPM specification for a one-to-one relationship between the OS and trusted platform; therefore, the TPM has a limitation in that it can be owned by only one entity at a time. If platform virtualization is used, the TPM must be virtualized to provide full TPM functionality for each guest entity. For this reason, specifications have been developed for a *Virtual TPM* (VTPM) (Berger, et al., 2006) to provide the TPM functions for each virtual environment on the platform.

There are several implementations of the VTPM; each uses different methodologies to virtualize the TPM. For LoBot we chose to use the Generalized VTPM framework the TCG is proposing (Scarlata, et al., 2008). In the framework, each VTPM is implemented as a software based emulator that has its own emulated resources including EK, SRK, PCRs, and NVRAM allowing it to function identically to a hardware TPM.

4.3 Trusted Execution Technology

Trusted eXecution Technology (TXT) is a feature of Intel's microprocessors and chipsets commonly referred to as Intel vPro (Grawrock, 2009). TXT provides a late launch capability for the secure creation and launch of virtual execution environments called *Measured Launch Environments* (MLEs). MLEs can be launched anytime after a platform is booted. Hardware protections are provided by TXT against software based attacks on MLEs during launch and while the MLE is executing.

Late launch is initiated through the SENTER function, which provisions the MLE on the TCP. The process measures and verifies the MLE with a secure hash algorithm built into TXT and correctly configures the processor and chipset prior to passing control of the processor to the MLE. This capability ensures that the MLE is the proper MLE the user wishes to use and that the MLE has been provisioned on the platform as intended.

5 Architecture

Information owners need the ability to manage their cloud datacenters to respond to the threats and balance their computation and network loads. The PVI architecture provides this capability through two layers that separate the security and management responsibility between the service provider and the client. The CVF layer provides computation resources managed by the service provider, while the PVI layer provides a virtual datacenter managed by the client. The service provider assumes responsibility for providing the physical security and the logical security of the service platform required for the PVI layer.

Clients are then able to configure their computing environment with the level of security required to meet their confidentiality and privacy requirements.

The primary requirement for the PVI architecture is protecting the AS where sensitive information is processing, which we call the *Application Domain* (aDom). To do so, we need to ensure that the aDom is provisioned exactly as intended, provisioned where intended, and provisioned when intended. To achieve this requirement, each TCFP in the cloud needs to be able to report security properties and configuration information to the PVI. These properties must be cryptographically bound and signed such that any authorized person can verify the properties. We use trusted computing techniques to provide cryptographically verifiable reports of the security properties and configuration of the TCFP.

A TCFP is the basic compute platform of the CVF. On each TCFP, multiple cloud VSs can be provisioned, along with VMs from other clients in the cloud; therefore, strong isolation is required on the platform. A VS is the basic component of processing within PVI. The VS consists of two sub-components, the aDom VM running applications with sensitive data, and its companion LoBot virtual appliance. Provisioning and migration require a management layer, which is controlled via the PVIF. Trusted storage and networking are also required components of the PVI; however, trusted storage and network issues are out of the scope of this paper.

5.1 Trusted Cloud Fabric Platform

A TCFP is one single physical machine in the CVF. Each TCFP implements Intel's vPro specification with TXT and a TPM version 1.2. The TPM implements a random number generator for nonces, encryption key generation, RSA 2048-bit encryption, and the SHA-1. Each TCFP may have multiple CPUs, but only has a single shared memory and a single TPM. A type 0 hypervisor is used to virtualize processing and most resources are shared amongst all virtual environments; however, local configuration could provide unshared resources for certain applications or domains. The platform can support many unprivileged domains with strong hypervisor isolation and hardware support via TXT.

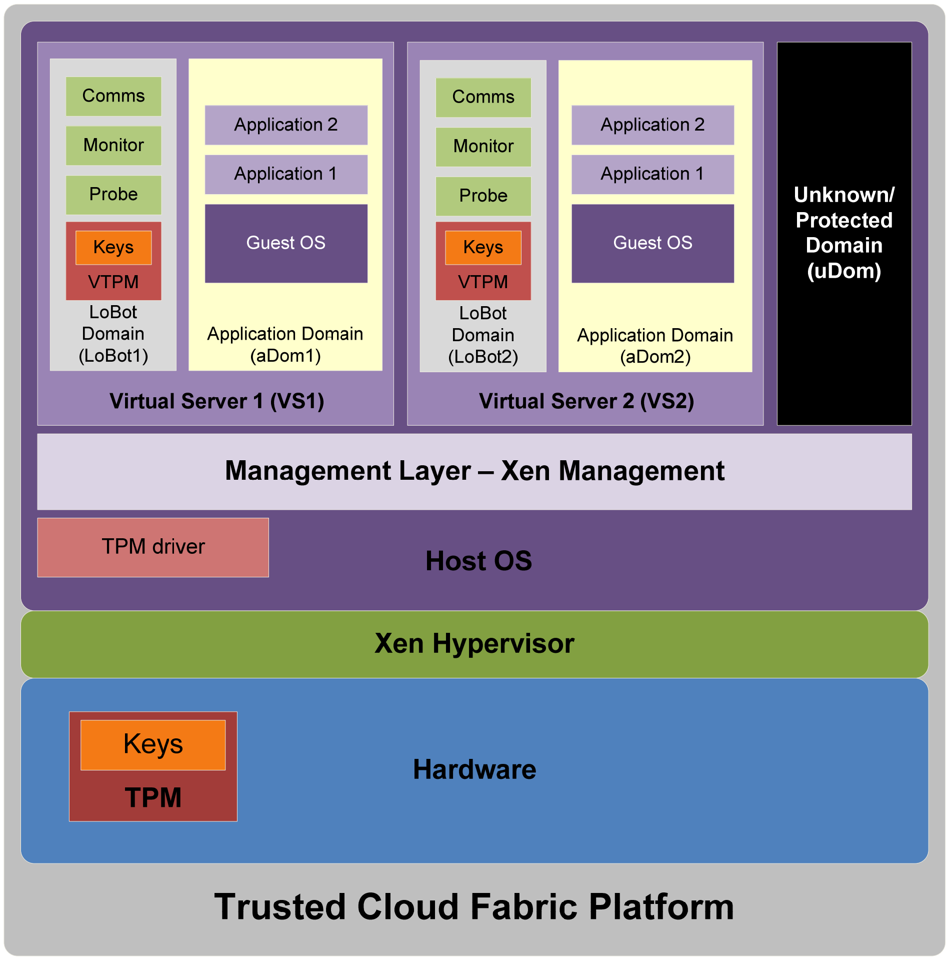

Figure 3 – Architecture of a Trusted Cloud Fabric Platform showing the Virtual Server Domain, Application Domain and LoBot Domain

The TCFP must have a *trusted computing base* (TCB) defined that can be measured via the TPM. We wish to have as small of a TCB as possible; however, there are certain components we wish to include in the TCB which make it rather large. At a minimum, the TCB should consist of the hardware components of the TCFP and the hypervisor. Additionally, we would like to include the kernel of the host OS in the TCB and possibly the management layer.

The TCFP node architecture shown in Figure 3 is developed around the Xen hypervisor (Barham, et al., 2003). The hypervisor runs at the highest privilege on the host platform; therefore, it must be a trusted and part of the TCB of the TCFP. The host operating system runs as a privileged domain (dom0), which provides the management layer for PVI; therefore, it is desired that dom0 be a trusted component as well.

The hypervisor and host OS can be verified through attestation of the host system by attestation from the host TPM's PCRs.

5.2 Management Layer

The management layer is responsible for controlling PVI provisioning and migration of VSs to trustworthy platforms within CVF. The management layer is a function of several components including XM and the PVIF. XM is used to manage local tasks on the TCFP while the PVIF manages task across multiple platforms throughout the entire PVI. Secure provisioning and live migrations are initiated through the PVIF and performed through the XM instances on the TCFPs. Secure provisioning and live migration are monitored and controlled by the PVIF via the *LoBot Secure Provisioning* (LSP) protocol.

The management layer must have privileges to read and write information into the TCFP's TPM to obtain information about the configuration of the TCFP and bind the LoBot's VTPM to the TPM. This requirement makes the management layer a trusted component of the architecture, which means that everything it is built upon must also be trusted.

PVI Factory

The PVI Factory serves as the controller, root authority, and policy decision point for the PVI. Figure 4 shows the overall PVI architecture with the PVIF as the management structure for PVI outside of the CVF. The PVIF is responsible for ensuring the integrity of the PVI and handling incidents in the event of a security breach. If any problems are detected, PVIF can shutdown the PVI, recall and inspect all images for tampering, and generate alarms and reports.

The PVIF is the most sensitive component of the PVI. The PVIF is the management controller and root of trust for PVI providing the following functions: virtual machine provisioning, VTPM key generation, EK certificate authority (VTPM Entity), and VTPM NVRAM storage. The PVIF is where all components of the PVI are provisioned and it is the root authority for provisioning, VTPM key generation, and certificate generation. Since VTPMs do not have valid EKs, an EK must be generated for each VTPM

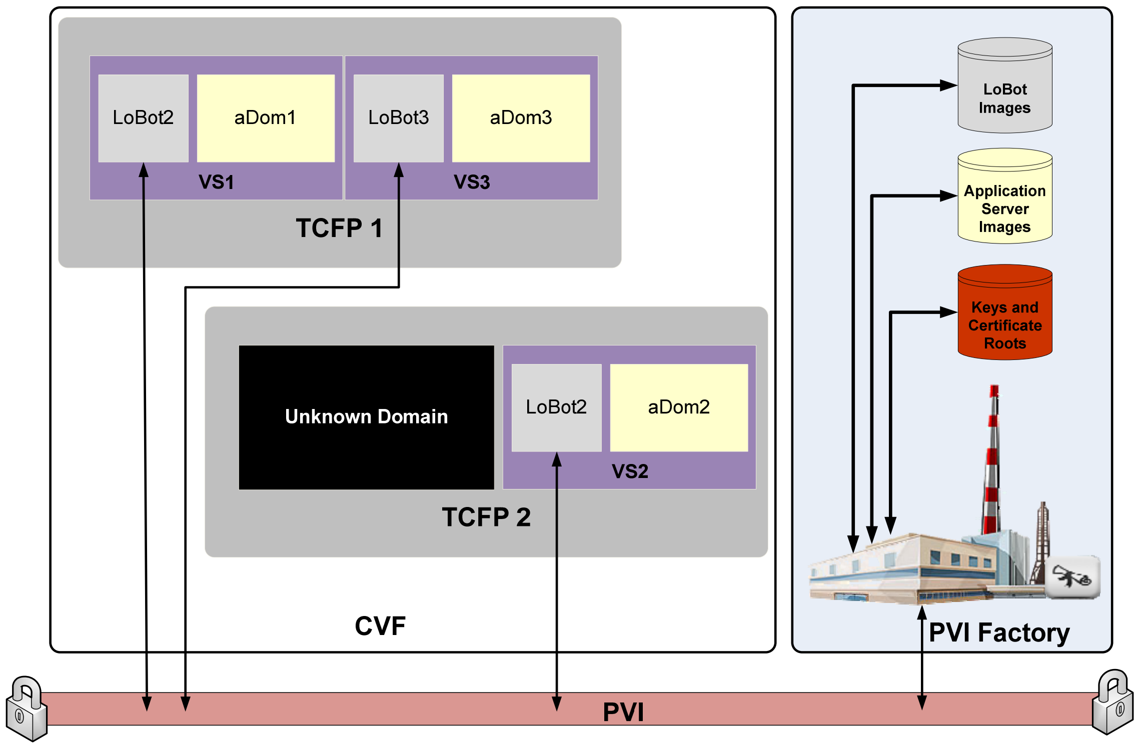

Figure 4 – PVI Architecture showing the PVI Factory and two TCFPs hosting three VSs and one unknown domain. The PVI Factory manages images and keys for the PVI and is the certificate authority.

used within the PVI to trust the operations of the VTPM. The factory also maintains master images for ASs, LoBots, and handles data transfers within the PVI through the VPN configuration and management.

The PVIF is an application that runs on a dedicated trusted server. Ideally, the PVI factory would be under full control of the information owner and not be collocated within the service provider's datacenter. Because the PVIF is the root authority, if it is compromised, then all existing PVI components are at risk of compromise and future provisioned components cannot be trusted. Therefore, the PVI factory should be under full control of the information owner, either as a standalone component in the datacenter or on the information owner's site. It should not be virtualized and should be isolated to the greatest extent possible from other systems. The PVI factory could have built-in hardware to accelerate cryptographic operations and to provide true randomization, but a software-only implementation would suffice for most applications.

5.3 Virtual Server Domain

Each *Virtual Server Domain* (vsDom) is a combination of an unprivileged aDom and a tightly coupled LoBot. The LoBot is implemented as an unprivileged stub driver domain to provide isolation and protection for the VTPM. VsDoms are provisioned, managed, and controlled by the client, not the operator, to ensure that the client maintains control of the information contained in the application server.

Application Domain

The aDom is the VM that the client runs application on in the cloud. This aDom is where all critical processing is performed and confidential data is processed; therefore, the domain needs to be protected at the highest level possible. ADom is the domain we wish to protect with PVI and LoBot. PVI's primary goal is protecting the aDom. Compromise of aDom would allow an attacker to gain access to or corrupt the sensitive data begin processed within aDom.

LoBot Domain

LoBot is a virtual appliance that is used to secure ASs in PVI. A LoBot is a self-contained VM image that has no external storage requirements, thus allowing it to move and replicate within a cloud environment efficiently. The LoBot domain's primary responsibility is to serve as the protector of the aDom. LoBot protects aDom through with the following primary functions: 1) emulate a physical TPM for the aDom and provide a root of trust for aDom, 2) act as a probe to measure target platforms, and 3) provide secure provisioning and live migration services. The probing function ensures the destination platform provides sufficient security properties to protect information of the aDom.

The LoBot architecture is a driver stub domain bound to a single aDom creating a vsDom. The LoBot provides TPM services via a VTPM to the aDom. The VTPM is the core of the LoBot architecture providing the same level of trust to the virtual platform as the TPM does on the physical platform. The LoBot's VTPM is bound to the physical TPM of the TCFP tightly coupling the LoBot and aDom to the host. Note PVI is owner of VTPM and CVF is owner of TPM. Placing the VTPM in the LoBot has several advantages over operating the VTPM in the host domain including isolation of the VTPM process and reducing the burden of VTPM migration. The VTPM instance is isolated in LoBot via Intel's Virtualization Technology (VT-x) and TXT such that no direct interactions with other domains are allowed. The only way to access the VTPM is through a rigidly controlled device interface between the LoBot and aDom ensuring that data stored by the VTPM is protected from other processes on the same platform. All information that a LoBot and VTPM are required to save, such as keys and non-volatile storage, is encrypted in a blob and sent to the PVI factory for storage. The PVI factory manages storing data for each LoBot eliminating the need for local storage.

Upon launch, the VTPM binds itself to the target's TPM then the probe application reads the platform configuration from the target TPM's PCRs and obtains identifying information about the platform. Identity information is provided in the form of cryptographic certificates. This information is then combined with the VTPM's PCR which is cryptographically sealed in a blob that is transferred to the PVI factory. The PVI factory decrypts the blob and examines the information received to make a trust decision. If PVI determines the target environment is trustworthy, the PVI factory configures the aDom image and securely transfers it to the target environment in a blob encrypted such that only the target platform may execute source environment.

At the target environment, the LoBot probe application receives and unseals the aDom image. If the image was tampered with during transfer, it will be detected during the decryption phase. To make sure everything is safe, the probe measures the source environment one more time to validate its integrity to ensure the launch in the target environment was successful.

5.4 Trusted Storage and Networking

Each aDom in PVI needs a virtual storage device to serve as the system disk and to store the software and data. Storage must be in a location that is accessible to both source and destination nodes during migrations. It is assumed the underlying management layer of the CVF will handle availability of the drives during migrations. In CVF, storage is implemented as drive images files stored on the SAN which emulates a block storage device. For our Xen prototype system, we used NFS4 with simple files as drive images. Storage must be trusted to maintain trust in the vsDom. For this paper, we assume virtual storage is trusted such that the confidentiality, availability, and integrity are all uncompromised. Further discussions on trusted storage are beyond the scope of this paper. Virtual storage issues are also out of the scope of this paper; however, storage migration may be necessary in the case a vsDom is migrated off of a network served by the SAN.

LoBots do not require block storage devices as they are non-persistent. This alleviates the need to have a trusted store for LoBots and ensures that the LoBot images are correct if the image is properly provisioned on the destination platform.

PVI operates on a high speed local area network inside CVF which is isolated from the Internet by firewalls, network address translation, and other security devices. To ensure private networking within PVI, encrypted virtual private networks and virtual LAN partitioning are used to isolate PVI from the other CVF network traffic. We assume the network boundary protection devices are robust enough to thwart attacks from the Internet including viruses, worms, and denial of service attacks.

6 The LoBot Protocols

The *LoBot secure provisioning* (LSP) protocols ensures that when a VM is provisioned within the PVI it is not subverted in any manner and the VM that is provisioned is exactly the same as the VM intended to be provisioned. LoBot is in concept a probe that ensures a safe provisioning by pre-measuring a target platform for trustworthiness. By sending a LoBot ahead of the aDom provisioning, the target platform can be measured to determine whether the target provides a trustworthy operating environment. This allows us to make a decision to provision to the target based on the results of the probe. Secure provisioning is achieved through the combination of determining a target platform's trustworthiness, ensuring that provisioning is not subverted in any manner, and verifying the VM is created and launched on the host platform as intended.

The LSP protocol can also be used for a live migration. Live migration is an extension of the secure provisioning protocol. The requirements of secure provisioning and secure live migration are the same:

Symbol	Meaning
S or SP	Source Platform
D or DP	Destination Platform
L or LoBot	Locator Bot
P or PVIF	PVI Factory
aDom	Application Domain
vsDom	Virtual server domain (aDom + LoBot)

Table 2 – Abbreviations used in Protocol

ensure a virtual machine is placed on the cloud platform as intended. Secure live migration is slightly more difficult than secure provisioning as there are three entities involved (PVI Factory, source and destination platforms) and the state of the source VM must be preserved to have a successful migration. In a live migration, LSP ensures that the VM's state is not compromised by the migration and that the VM migrated to the new platform resumes operation at the point of suspension on the originating machine.

We created a new management application on the PVIF called *pvif* that is used to initiate the provisioning and live migration LSP protocols. When *pvif* is initiated with a *create* subcommand, a secure provisioning is initiated. The *migrate* subcommand initiates the secure live migration. The protocol uses the symbols shown in **Error! Reference source not found.**.

6.1 Secure Provisioning Protocol

The following steps describe the secure provisioning of a VS (both aDom and LoBot) from the PVI Factory (PVIF) to a destination platform (DP) machine using the LSP protocol in detail.

1. PVIF initiates a provisioning of vsDom on the destination platform (DP) via:

PVIF: pvif create vsDom DP

The PVIF starts the process of provisioning vsDom, which consists of aDom and associated LoBot

2. PVIF requests an AIK certificate from DP.

$$PVIF \leftarrow DP\colon\ cert\left(AIK_{pubDP}\right)$$

The AIK is used to verify that DP has a valid TPM and determine whether that TPM is a real or virtual. This step also verifies that the TPM has a feature set which is acceptable to the PVIF. Note that we chose to use the AIK in this step to preserve privacy of the CVF; however, an EK could be used if privacy is not a concern of the CVF operator or the AIK is not available.

3. PVIF creates a non-predictable 160-bit nonce and sends it to DP requesting a quote of the AIK.

$$PVIF \rightarrow DP\colon\ nonce$$

4. PVIF requests DP's SRK public key, which is sent along with the nonce signed with DP's private AIK.

$$PVIF \leftarrow DP\colon\ sign\left\{SRK_{pubDP}, nonce\right\}AIK_{privDP}$$

PVIF then verifies the nonce is the same as previously sent and that the signature is validated by the CA that created the AIK.

5. PVIF creates a *migratable storage key* (*MSK*), wraps it with DP's SRK, and sends to DP. PVIF then generates the EK for the LoBot's VTPM, encrypting it with the MSK so that is will be bound to the DP ensuring no other entity can use it. The encrypted EK_L is then placed in LoBot image.

$$PVIF \rightarrow DP\colon\ SRK_{pubDP}\{MSK_{DP}\}$$

$$PVIF\colon\ MSK_{DP}\{EK_L\} \rightarrow LoBot$$

6. PVIF then encrypts the entire LoBot image using the SRK public key and a symmetric encryption key (K_1), and sends the entire package to DP.

$$PVIF \rightarrow DP\colon\ SRK_{pubDP}\left\{K_1\{LoBot\}\right\}$$

7. DP receives the encrypted LoBot package and decrypts it using its SRK private key and K_1.

$$DP\colon\ SRK_{privDP}\big\{K_1\{LoBot\}\big\} \rightarrow LoBot$$

If LoBot is successfully decrypted, then it is launched by DP via the late launch capability of the TXT SENTER function, creating a MLE that verifies the LoBot is launched correctly.

8. The newly launched MLE measures LoBot, storing the results in the LoBot's vTPM's PCRs 0-5 and extends DP's PCR 6 with the measurements. The LoBot's PCR 6 is extended with DP's PCR(0-6).

$$LoBot\colon\ extend\{PCR_{DP}(6), hash(PCR_L(0-5))\}$$

$$LoBot\colon\ extend\big\{PCR_L(6), hash\big(PCR_{DP}(0-6)\big)\big\}$$

9. LoBot quotes DP's PCRs and its own PCRs to PVIF

$$LoBot \rightarrow PVIF\colon\ sign\{PCR_{DP}(0-15), PCR_L(0-6)\}AIK_L$$

10. PVIF determines trustworthiness of DP based on measurement returned from LoBot.
    10a. If DP is deemed trustworthy, PVIF continues provisioning by initiating transfer of app domain.
    10b. If DP is deemed untrustworthy, PVIF notifies LoBot to destroy itself and terminates provisioning.

11. PVIF hashes aDom and encrypts aDom and hash value with K_2. PVIF then wraps K_2 with the LoBot's SRK and PCR(0-6) to ensure only the LoBot can unseal and the state of the DP has not changed. Note that PVIF has LoBot's keys and the ability to emulate a TPM enabling PVIF to seal images for LoBot.

$$PVIF \rightarrow LoBot\colon\ \big(SRK_L, PCR_L(0-6)\big)\{K_2\{hash\{aDom\}, aDom\}\}$$

12. LoBot unseals aDom and the hash measurement, hashes aDom and compares it to the sent hash to determine the transfer success.

$$LoBot\colon\ \left(SRK_L, PCR_L(0-6)\right)\{K_2\{aDom\}\} \rightarrow aDom$$

$$LoBot\colon hash\{aDom\} == \left(SRK_L, PCR_L(0-6)\right)\left\{K_2\{hash\{aDom\}\}\right\}$$

13. If the hashes compare, aDom is launched via SENTER. If SENTER succeeds, a safe provisioning occurred; otherwise, the LoBot reports unsuccessful migration to PVIF, destroys the aDom and then itself.

6.2 Secure Live Migration Protocol

A live migration can be initiated at the source platform or PVI factory, but to maintain trust in PVI, the PVI Factory manages the live migration. The following steps describe live migration with the probe protocol of a VS (both aDom and LoBot) from a source platform (SP) to destination platform (DP).

1. Initiate a live migration from the source platform to the destination platform on PVIF via the following command:

   PVIF: pvif migrate domA1 DP

   In the event the source platform needs to initiate the migration, the command must be sent to the PVIF to start the migration.

2. PVIF requests an AIK certificate from DP to verify DP's TPM.

$$PVIF \leftarrow DP\colon\ cert\left(AIK_{pubDP}\right)$$

   PVIF already knows the AIK of SP since it would have had to previously provision it.

3. PVIF creates a non-predictable nonce and sends it to both SP and DP.

$$PVIF \rightarrow SP, DP\colon\ nonce$$

4. PVIF requests DP's SRK public key be sent to SP, which is sent along with the nonce signed with DP's private AIK.

$$SP \leftarrow DP\colon\ sign\{SRK_{pubDP}, nonce\}AIK_{privDP}$$

SP then verifies the nonce is the same as previously sent and that the signature is validated by the CA that created the AIK. SP reports nonce back to PVIF.

5. PVIF requests P to halt aDom, capture its state, and begin migration process. SP then clones the LoBot and rewraps the MSK_{SP} for the VTPM with the public SRK from DP.

$$SP \rightarrow DP\colon\ SRK_{pubDP}\{MSK_{DP}\}$$

6. SP encrypts $LoBot_S$ image and sends to DP.

$$SP \rightarrow DP\colon\ SRK_{pubDP}\{K_1\{LoBot_S\}\}$$

7. DP receives encrypted LoBot blob and decrypts it using its SRK private key and symmetric key K_1.

$$DP\colon\ SRK_{privDP}\{K_1\{LoBot_S\}\}$$

If LoBot is successfully decrypted, then it is launched by DP via the late launch capability of the TXT SENTER function, creating an MLE. $LoBot_S$ becomes $LoBot_D$.

8. The MLE measures $LoBot_D$, storing the results in the LoBot's vTPM's PCRs 0-5 and extends DP's PCR 6 with the measurements. The LoBot's PCR 6 is then extended with DP's PCR(0-6).

$$LoBot_D\colon\ extend\{PCR_{DP}(6), hash(PCR_L(0-5))\}$$

$$LoBot_D\colon\ extend\{PCR_L(6), hash(PCR_{DP}(0-6))\}$$

9. $LoBot_D$ quotes DP's PCRs to PVIF and its own PCRs to $LoBot_S$.

$$LoBot_D \rightarrow PVIF\colon\ sign\{PCR_{DP}(0-15)\}AIK_L$$

$$LoBot_D \rightarrow LoBot_S\colon\ sign\{PCR_L(0-6)\}AIK_L$$

10. PVIF determines the trustworthiness of destination based on measurement returned from LoBot.

    10a. If DP is deemed trustworthy, PVIF notifies SP to continue transfer aDom.

    10b. If DP is deemed untrustworthy, PVIF notifies LoBot to destroy itself and terminates migration.

11. Source binds measurement with LoBot's PCRs and sends to LoBot.

$$LoBot_S \rightarrow LoBot_D\text{:}\ \ \big(SRK_L, PCR_L(0-6)\big)\big\{K_2\{hash\{aDom\}, aDom\}\big\}$$

12. $\mathrm{LoBot_D}$ measures aDom, unseals measurement and compares its measurement to the sent measurement.

$$LoBot_D\text{:}\ \ \big(SRK_L, PCR_L(0-6)\big)\big\{K_2\{aDom\}\big\} \rightarrow aDom$$

$$LoBot_D\text{: } hash\{aDom\} == \big(SRK_L, PCR_L(0-6)\big)\ \Big\{K_2\{hash\{aDom\}\}\Big\}$$

13. Determine migration success.

    13a. If the hashes compare, aDom is launched via SENTER. If SENTER succeeds, a safe migration occurred; $\mathrm{LoBot_D}$ reports successful migration and $\mathrm{LoBot_S}$ destroys aDom and itself on SP.

    13b. If the hashes do not compare or SENTER fails, $\mathrm{LoBot_D}$ reports unsuccessful migration to SP, SP unsuspends operation of LoBot and aDom, and reports failure. $\mathrm{LoBot_D}$ destroys aDom and itself on DP.

7 Discussion

Improving the overall security of the cloud is the ultimate contribution of PVI and LoBot. The PVI architecture creates the shared security posture necessary to manage the virtual datacenter in the cloud and securely isolates the virtual datacenter from malicious actors, other applications, VMs, and malware (*e.g.*, viruses and worms), enabling the data owner to use commodity internet resources and reduce their overall IT overhead.

LoBot and the LSP protocols provide high-assurance mechanisms to provision and migrate virtual machines securely in the cloud. LoBot uses VTPMs to provision trustworthy VMs in the cloud requiring individual computing platforms within the cloud to have a TPM owned accessible by LoBot. The LoBot's trust authority for the VTPM is the PVI Factory and linking the VTPM to the platform's physical TPM creates a dual rooted trust for the application domain. This dual rooted trust anchors the application domain to the host platform and PVI preventing tampering and cloning and ensuring the data is protected from adversaries in the cloud.

The LSP protocol provides an assured mechanism to provision a VM securely on a destination platform. There are several advantages of this protocol over existing provisioning techniques. Existing provisioning protocols do nothing to assure the safety and integrity of the destination environment before the provision occurs and little to ensure that the provision occurred safely. The existing protocols have fallback mechanisms in the event of a failed transfer, but once a VM is placed in a malicious or unsafe environment, the VMs confidential data is compromised and cannot be recovered.

The LSP protocol provides two assurances that were previously unavailable: first, the target platform we wish to provision on is a platform we trust (e.g. it has a configuration and security properties that are acceptable) and second, we know if the VM provisioning or migration occurs successfully. The trust decision is made by the PVIF from via the LoBot pre-measuring the platform reading by reading the PCRs of the target's TPM and measuring other properties of the target platform. If target platform is demed trustworthy, we continue with the provisioning as planned; otherwise, we back out of provisioning preventing data in the VM image from being sent to an untrustworthy machine. The only risk of contacting a malicious server is an attacker would be able to obtain some information and keys from a LoBot, but since LoBot has a minimal set of functions, the risk is insignificant. If the LoBot is compromised, all keys and references to that individual LoBot must be destroyed to ensure that no compromises can be achieved via the LoBot's credentials. Determining if a VM provisioning occurred as intended and not subverted in any manner is accomplished by cryptographically binding the VM image to the target configuration ensuring that the VM cannot be provisioned anywhere other than the intended target and the configuration of the target is not be altered between measurement and provisioning.

From the above analysis, we can see that the LSP protocol defends against all three of the attacks discussed earlier. The malvm and malhost are detected by the LoBot during initial probe phase. LoBot detects if a machine is not registered, has an invalid certificate, or has an alternate configuration from what is expected, and prevents the VM from provisioning on the machine. A man-in-the-middle attack is thwarted by the protocol through encryption of the VM image and tampering is detected by the hash comparisons.

A current limitation of the LSP protocol is the vulnerability of compromise to host machine after provisioning. Changes to the node's configuration are not detected after the initial trustworthiness of the node is determined. If the attacker gains control of the host machine after provisioning, the CVF architecture could be compromised. The PVI architecture and protocol are designed to protect the application server from being migrated to an environment that is predetermined to be malicious. We are exploring monitoring capabilities for LoBot that would detect changes to the host machine post provisioning.

PVI's security properties are in addition to any link encryption, secure tunneling, virtual private networking, virtual LANs, and other techniques used to protect the virtual datacenter. PVI is another layer of a good defense in depth strategy allowing us to achieve a very high security posture within the cloud and provide a high level of assurance that sensitive information is not being comprised.

8 Further Work

The current development on PVI is mostly "proof of concept" prototypes. To make a viable product the protocol needs to be integrated into the Xen management infrastructure and the PVI factory needs to be fully implemented. A number of details still remain to be worked out, specifically what are the proper properties required for pre-measurement that actually make sense in terms of reporting the security properties of a platform. Beyond the basic features of LoBot, we would like to add additional functionality such as continuous monitoring and sensor net like capabilities for detecting malicious activities.

This work is part of a larger project to build trust and security mechanisms for VMs. The Virtual Centurion project at the UMBC Cyber Defense Lab is a series of technologies enabling trustworthy virtualized platforms to increase the security of applications such as cloud computing, mobile computing, and voting. We are developing new trust mechanisms for cloud computing applications. Although an industry standard VTPM implementation upon which we can further build and refine the LSP protocol will soon be a reality, we believe additional extensions would improve trust in a cloud computing environment. We are also working on uniquely identifying trusted virtual machines through cryptographic binding of identity information with the VTPM and LoBot. With the combination of PVI, LoBot, trust mechanisms, and trusted virtual machine identification we have a very powerful set of technologies on which to build trusted virtual infrastructures in the cloud.

9 Conclusion

This paper proposes a new paradigm for securing and managing cloud computing services based on a synergistic relationship between the vendor and customer of cloud services. This relationship provides an increased security posture while allowing both parties to set security controls required to protect the infrastructure and data within the cloud and virtual datacenter.

Cloud computing service providers need to enable a transparent view of their infrastructure so their customers can understand the security posture and threats to the system. This capability will give the vendor a competitive advantage as a secure system provider over vendors who choose to obscure their infrastructure inner workings to protect proprietary technology. Cooperation between vendor and customer will result in increased security while lowering the overall cost of ownership for IT infrastructure.

We have shown how to provision and migrate virtual machines securely in a cloud computing environment via the LoBot Secure Provisioning protocol. The architecture and protocol implement our new security concepts for securing virtual datacenter in the presence of threats. Data owners can verify configuration and security settings of target platforms by probing the fabric with a LoBot to obtain the destination environment's properties increasing the assurance that their VMs will not be compromised. Private Virtual Infrastructure increases security and privacy in the cloud by isolating the virtual datacenter from the greater cloud.

We have shown that using trusted computing technologies in the cloud computing environment can benefit both operators and clients. The TPM and VTPM provide the root of trust and security needed by the protocol and Intel TXT provides the late launch capability needed to securely launch a VM.

This is the first protocol that we are aware of for provisioning and migrating live virtual machines that first tests the target environment to ensure it meets policy and configuration requirements before the actual environment is relocated. Additionally, it provides assurance that the transfer is not tampered with during the process. The protocol is a preemptive measure to ensure that critical data is not exposed to malicious actors without a means for recovery.

Security is the responsibility of all parties involved in utility cloud computing. Vendors are responsible to provide a secure fabric. Information owners are responsible to protect their data. PVI provides information owners the flexibility to manage their own data while realizing the cost benefits of cloud computing.

Acknowledgment

We thank Russell Fink, John Pinkston, Brooke Stephens, and David Challener for their helpful comments.

References

TCG Specification Architecture Overview: Trusted Computing Group; August 2 2007.

TPM Specification Version 1.2 Revision 103: Trusted Computing Group; July 9 2007.

Barham P, Dragovic B, Fraser K, et al. Xen and the Art of Virtualization. *ACM SIGOPS Operating Systems Review*. 2003;37(5):164-177.

Berger S, Cáceres R, Goldman KA, Perez R, Sailer R, van Doorn L. vTPM: Virtualizing the Trusted Platform Module. *Proceedings of the 15th USENIX Security Symposium*. Vancouver, B.C.; 2006.

Berger S, Cáceres R, Pendarakis D, et al. TVDc: Managing Security in the Trusted Virtual Datacenter. *ACM SIGOPS Operating Systems Review*. January 2008;42(1):40-47.

Bussani A, Griffin JL, Jansen B, et al. *Trusted Virtual Domains: Secure Foundations for Business and IT Services*. Yorktown Heights, NY: IBM; November 9 2005. RC23792.

Carr NG. *The Big Switch: Rewiring the World, from Edison to Google*. New York: W.W. Norton & Company; 2008.

Chen L, Landfermann R, Löhr H, Rohe M, Sadeghi A-R, Stűble C. A Protocol for Property-Based Attestation. *Proceedings of The First ACM Workshop on Scalable Trusted Computing*. Fairfax, VA: ACM; 2006.

Gallery E, Nagarajan A, Varadharajan V. A Property-Dependent Agent Transfer Protocol. In: Chen L, Mitchell C, Martin A, eds. *Second International Conference on Trusted Computing*. Vol 5471/2009. Oxford: Springer; 2009.

Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D. Terra: A Virtual Machine-Based Platform for Trusted Computing. *Proceedings of the 19th ACM Symposium on Operating Systems Principles*. Bolton Landing, NY: ACM; 2003.

Gasmi Y, Sadeghi A-R, Stewin P, Unger M, Asokan N. Beyond Secure Channels. *Proceedings of the 2007 ACM Workshop on Scalable trusted computing*. Alexandria, VA: ACM; 2007.

Gibson S. ARP Cache Poisoning. [Webpage]. Dec 11, 2005; Available at: http://www.grc.com/nat/arp.htm. Accessed March 20, 2009.

Gibson S, Laporte L. BailiWicked Domain Attack. *Security Now* [Podcast]. No. 155. July 31, 2008; Available at: http://www.grc.com/securitynow.htm. Accessed June 15, 2009.

Goldman K, Perez R, Sailer R. Linking Remote Attestation to Secure Tunnel Endpoints. *First ACM workshop on Scalable Trusted Computing*. Alexandria, VA: ACM; 2006.

Grawrock D. *Dynamics of a Trusted Platform*. Hillsboro, OR: Intel Press; 2009.

Krautheim FJ. Private Virtual Infrastructure for Cloud Computing. *Workshop on Hot Topics in Cloud Computing*. San Diego, CA; 2009.

McCune JM, Jaeger T, Berger S, Cáceres R, Reiner S. Shamon: A System for Distributed Mandatory Access Control. *Annual Computer Security Application Conference*. Miami Beach, FL; 2006.

Murray DG, Milos G, Hand S. Improving Xen Security through Disaggregation. *Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. Seattle, WA: ACM; 2008.

Nurmi D, Wolski R, Grzegorczyk C, et al. *Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems*. Santa Barbara, CA:

University of California, Santa Barbara Computer Science; October 2008. Technical Report 2008-10.

Pearson S. *Trusted Computing Platforms: TCPA Technology in Context*. Upper Saddle River, NJ: Prentice Hall; 2003.

Sadeghi A-R, Stüble C. Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms. *2004 Workshop on New Security Paradigms*. Nova Scotia, Canada: ACM; 2004.

Sailer R, Valdez E, Jaeger T, et al. *sHype: Secure Hypervisor Approach to Trusted Virtualized Systems.* Yorktown Heights, NY: IBM; 2005. RC23511.

Sailer R, Zhang X, Jaeger T, van Doorn L. Design and Implementation of a TCG-based Integrity Measurement Architecture. *13th USENIX Security Symposium*. San Diego, CA; 2004.

Santos N, Gummandi KP, Rodrigues R. Towards Trusted Cloud Computing. *Workshop on Hot Topics in Cloud Computing*. San Diego, CA; 2009.

Scarlata V, Rozas C, Wiseman M, Grawrock D, Vishik C. TPM Virtualization: Building a General Framework. In: Pohlmann N, Reimer H, eds. *Trusted Computing*. Wiesbaden, Germany: Vieweg+Teubner; 2008:43-56.

Vaquero LM, Rodero-Merino L, Caceres J, Lindner M. A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review.* 2009;39(1):50-55.