RZ 3655 (# 99675) 08/21/2006
Computer Science 15 pages

# Research Report

## An Open Trusted Computing Architecture — Secure Virtual Machines Enabling User-Defined Policy Enforcement

Dirk Kuhlmann

HP Laboratory
Bristol, UK

Rainer Landfermann

Ruhr-University
Bochum, Germany

HariGovind V. Ramasamy, Matthias Schunter

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Gianluca Ramunno, Davide Vernizzi

Politecnico di Torino
Italy

# An Open Trusted Computing Architecture — Secure Virtual Machines Enabling User-Defined Policy Enforcement

Dirk Kuhlmann[*], Rainer Landfermann[†], HariGovind V. Ramasamy[‡], Matthias Schunter[‡], Gianluca Ramunno[§], Davide Vernizzi[§]

June 28, 2006

**Abstract**

Virtualization of computers enables a wide variety of applications ranging from server consolidation to secure sandboxing of malicious content. Today, lack of security of virtual machines is a major obstacle for broad adoption of virtual machine technology. We address this obstacle by an open architecture that adds scalable trusted computing concepts to a virtual machine infrastructure. The platform has a layered system architecture, and from bottom to top consists of a Trusted Platform Module (TPM) specified by the Trusted Computing Group (TCG), a trusted virtualization layer with strong isolation properties (among virtual machines) and well-defined interfaces to the TPM, and security services (such as protected storage, security policy enforcement, and identity management). We describe the guiding principles and the overall architecture of the platform, and detail the advantages of such an architecture. The platform can be leveraged to significantly enhance the security and trust properties of the standard operating systems, middleware, and applications hosted atop the platform. We believe the platform has wide-ranging applicability particularly in the context of distributed scenarios with inherent, multilateral trust and security requirements. We give examples of such scenarios that would be enabled by the platform.

## 1 Introduction

The task of securing personal computer (PC) platforms is becoming increasingly difficult. The increasing mobility of clients between networks and the ubiquity of networks have contributed to rapid exploitation of security vulnerabilities. Addressing this issue requires resolving an assortment of challenges including:

- protection of personally identifiable data against malicious disclosure by Trojan horses or viruses,
- enabling the user to determine whether a personal workstation is in a secure state or not,
- protection of cryptographic keys for authentication and confidentiality, and
- enforcement of security policies even if parts of the machine may be under attack or are already compromised.

[*]HP Labs, Bristol, UK
[†]Ruhr-Universitt Bochum, Germany
[‡]IBM Research, Zurich, Switzerland
[§]Politecnico di Torino, Italy

Software-based techniques can address parts of these problems if the core operating system functions correctly. Unfortunately, most smart attacks target the core operating system and the deployed security software. Thus, techniques implemented in software-only will have limited effectiveness.

Our approach to address the above issues and limitations is to combine two core technologies: (1) *Virtual Machines*[1], which allow containment of attacks and scoping of trust, and (2) *Trusted Computing*, which provides secure hardware that can act as a core root of trust. This combination enables remote verification and local fallback security if software is compromised.

While virtualization is attractive for improving security through virtual machine isolation, as Garfinkel and Rosenblum [GR05] rightly point out, it opens a whole new pandora's box of security problems that the state-of-the-art in virtualization technology hardly addresses. The specific technical challenges that result when combining those two technologies are:

1. providing an infrastructure with a set of services that implement scalable security for virtual machines,

2. hardening the virtualization software with the goal of providing an isolation degree among virtual machines that is as close as possible to the isolation among physical machines.

3. leveraging Trusted Computing technology (e.g., for attesting to the integrity of the virtualization layer) while at the same time providing a choice of acceptable policies (e.g., satisfying privacy concerns) to the users.

The last point deals with the enforcement of *multilateral security* requirements. Enforced security policies need to balance the security requirements of all stakeholders.

We describe a platform that is built upon the foundation of the hardware root of trust offered by the Trusted Platform Module (TPM) and leverages the recent advances in hardware virtualization such as virtualization support in the CPU offered in latest chips from Intel and AMD. The platform called OpenTC [Ope] (which stands for Open Trusted Computing) is layered and consists of actual hardware, a trusted virtualization layer with strong isolation properties (among virtual machines), and security services (such as protected storage, TPM services, key management, security policy enforcement, and identity management).

Unlike existing proprietary point solutions in the Trusted Computing space, our platform is based on open design and open-source security technology. This allows users to adapt the platform as needed while verifying the platform's security policies and mechanisms. A prime motivation for an open design and open-source-based approach is to build a trusted platform that can illustrate the benefits of Trusted Computing for each individual end-user. In particular, the OpenTC platform cannot be used to enforce policies that a user has not consented. Furthermore, the owner of a PC would be able to accept different security policies for each of the virtual machines running on the physical PC. For example, while a virtual machine holding personal details of an user can be configured with a privacy focus, another virtual machine co-located on the same physical PC and owned by the user's employer may have more of a security focus.

*Outline:* In Section 2, we survey our security and design requirements. In Section 3, we survey related work. In Section 4, we outline our architecture. In Section 5, we describe our usage scenarios. In Section 6, we conclude and identify open questions. Appendix A outlines the threat model and the proposed countermeasures for a secure transaction usage scenario.

[1] When we talk about virtual machines in the context of this paper, we are interested in multiple virtual personal computer instances on a single piece of hardware.

## 2 Design Goals

In this section, we explain the principles that guide our design and implementation.

*Open Specification, Implementation, and Validation:* Trustworthiness requires open specification, implementation, validation, and execution. Due to its very nature, the Trusted Computing approach requires powerful security mechanisms at a very low system level. These mechanisms can affect the execution of every other system component. We reduce the risk that software violates the interests of system owners by allowing the inspection of both design and implementation. Furthermore, we provide different implementations of core functionality to validate interoperability. One example is the deployment of our security services on two different hypervisors.

*Explicit Policies Separated from Mechanisms:* Policies should be explicit and separated from mechanisms. Policies are only enforced if if they are in accordance with the subject's explicit consent. This principle should also be honored by protecting the user's execution environment against unilateral interventions of other parties such as system operators or content owners. However, in enterprise scenarios unilateral intervention must be supported for the benefit of a larger system. If a user has consented to a policy of its employer, audit trails should be provided to support ex post dispute resolution.

*Power to the User:* User control should be maximized. Today, Trusted Computing only provides a choice to turn off the Trusted Platform Module and thus disable desirable benefits, too. OpenTC aims at providing finer-grained choices. For example, a user may choose to add a virtual TPM device to a virtual machine that only exposes a subset of the TPM functionality. As a general rule, this principle is equally applicable to operating systems and other software components: software must not inhibit the system owner from executing the right to opt out at any given stage. This means that, for example, an individual should be able to just delete an enterprise partition to opt out of the associated policies.

*Multilateral Security and Privacy:* Multilateral policies should be negotiable. Trusted Computing mechanisms can be used to impose temporary, functional restrictions on execution environments that enforce mutual agreements between the system owner and other parties. These agreements can be conceptualized as 'transaction security contracts.' They concern technical configurations and are enforced by technical means. Note that these important differences to real world contracts has to be taken into account. Another important aspect is privacy protection for individuals. In particular for transaction security contracts, the identity of a user as well as other personally identifiable information is often not relevant. As a consequence, these secure transactions should be enabled without revealing any personal information.

*Scalability:* In order to be robust and scalable, data must be migrateable. Software migration should be enabled between platforms with equivalent policies and protection levels. Requirements of securing platforms and protecting data have to be balanced against those of service continuity, manageability, and product life-cycles, and fair sharing. Technical options of binding data and software to particular trusted platforms must be complemented by mechanisms allowing to migrate and share protected elements.

## 3 Related Work

**Virtual Machine Monitors** Our aim is to build a framework for running secure and trusted operating systems. From the security perspective we rely upon two existing and open source virtualization technologies, Xen [BDF+03] and Fiasco [Grob, Hoh98], used to guarantee the isolation among applications, services or fully fledged operating systems – the so called compartments or domains – that run concurrently.

Xen is a virtual machine monitor or hypervisor that supports the execution of multiple guest operating systems. Xen 2 used a para-virtualization approach to achieve better performances than the traditional approaches with a full host system, but required a modified

kernel. Xen version 3.0 also supports full virtualization of the guest machines to run unmodified operating systems. Xen 3 requires hardware support for virtual machine monitors provided by the next generation of AMD and Intel processors.

Fiasco is a small and efficient microkernel implementing the L4 API that concurrently executes different tasks and guarantees the isolation between them through control of interprocess communications. L4 can run Linux as one of these tasks [HHL+97]. Again, this either requires a modified guest operating system or else hardware support.

Many other virtualization technologies have been proposed and implemented. In [GPC+03] the authors proposed Terra, a Virtual Machine Monitor able to perform the binary attestation of virtual machines in software in two different flavours: 'ahead-of time' for small VMs and 'optimistic' for large VMs. Note that Terra does not use secure hardware.

VMware [MN00] is a commercial software that implements the full virtualization on top of host Operating System (Linux and MS-Windows). A version also exists capable to run the virtual machine monitor on the "bare metal" like Xen does: VMware ESX server [vmwa] is able to run wide set of platforms. Hardware support for virtualization is still an experimental feature in VMware ESX, but it is planned to be fully supported in next versions [vmwb].

**Trusted Computing** The specification of the Trusted Platform Module has been standardized by the Trusted Computing Group (TCG). The TPM is tamper-resistant and provides core security functions (such as random number generation, computation of cryptographic hash, RSA key generation, RSA signature generation/verification, and storage of keys) in hardware. Our architecture is meant to be deployed on TPM-equipped devices (such as PCs, servers, mobile phones) and leverages the TPM as a trusted third party for interactions between software hosted atop the platform and outside distributed applications. Already, TPM chips are widely deployed – many notebook and desktop manufacturers now ship their products with TPM chips.

TPM-enabled boot loaders have been proposed in [Groa, MSMW03] in order to store the measures of the loaded software modules. The Enforcer [MSMW03, MSWM03] is a Linux security module that adds Tripwire-like features and uses the TPM for dynamic integrity checking of the file system and for securely storing the password of the encrypted loop device. An integration of SElinux and the previous TPM-based Enforcer has been proposed in [MSW+04].

[SZ03] only uses TPM to sign the system and verify signatures to discover attacks, but does not use any other TPM features like remote attestation. [SvW04] introduces remote attestation, but differ from our architecture because it does not consider privacy issues and isolated execution which are taken into account by our architecture. In [MSW+04] the system proposed uses TPM and is capable of being attested, but provide isolation between different applications using SELinux policies and not a virtualization layer.

**Secure Operating Systems** The use of virtualization techniques is one possibility for enhancing operating system security. Other approaches are Linux with enhanced access control policies such as SELinux [NSA]. Unlike our architecture, SELinux does not use trusted computing to enhance security.

The Next-Generation Secure Computing Base (NGSCB) [CJPL02], formerly Palladium, is a Microsoft concept for a Trusted Operating System based on virtualization and Trusted Computing hardware. It seems as if only a minor fraction of its concepts (such as harddisk encryption [Mica] but not virtualization) are transferred into the actual Vista [Micb] product.

**Attestation** One of the key feature that the TPM enables is remote attestation. This allows a remote stakeholder to verify the integrity of a machine. Attestation has been examined by several projects [MSMW03, SvW04, SZ03]. The goal is to enhance Linux with support for integrity measurement and binary attestation.

Binary attestation can be a straighforward way to verify the integrity of the platforms, but it suffers some problems. Lack of scalability is a major drawback of binary attestation: binary attestation requires the verifier to know all possible hash-values of all OS compo-

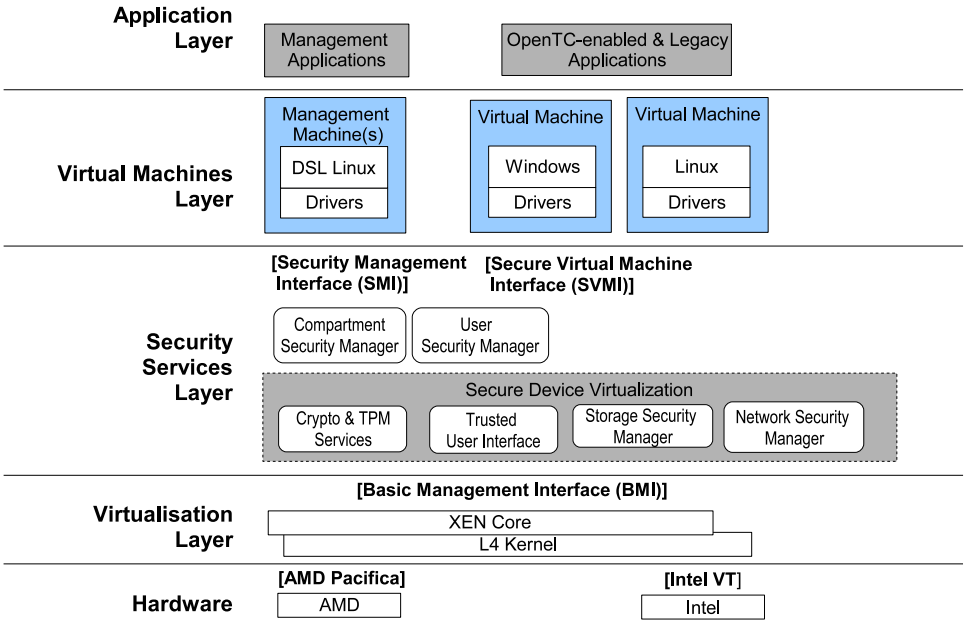


Figure 1: Layers of the OpenTC Architecture

nents of any machine that he may want to verify. Another concern is privacy: by revealing the configuration of the system to be attested, the binary attestation causes an unwanted leakage of private information. We pursue a different approach called property-based attestation [PSHW04, SS05]. Through this approach, a verifier can validate properties of a remote platform without receiving the related configuration details. Example properties that can locally be verified are the version and distribution used to install a machine, the fact that all installed software is authorized by the owner of the machine, the owner is a member in a trusted list of users, and the machine is properly isolated from untrusted networks. The core idea is to use a plugin to perform these checks while using binary attestation to validate the plugin and authenticating its output.

# 4 The Open Trusted Computing Architecture

Figure 1 outlines our architecture. The unique features of the OpenTC architecture are:

- Verifiable security by means of trusted computing technology.
- Support of multiple different hypervisors (L4 and Xen).
- Flexibility by means of configurable policies.

It is structured in different layers of abstraction that we will describe in the sequel. Each layer interact with the next layer of abstraction by a set of well-defined interfaces.

The foundation of our architecture is actual virtualization-enabled x86 processor and its peripherals. This includes processors, memory, and devices (network, storage, PCI cards, etc.) that need to be virtualized. The hypervisors used support AMD SVM technology[2] as well as Intel VT technology[3] By using processors with full virtualization support, we can achieve better isolation without the need to modify guest operating systems.

[2]See `http://enterprise.amd.com/us-en/solutions/consolidation/virtualization.aspx`
[3]See `http://www.intel.com/technology/computing/vptech/`

## 4.1 Virtualization Layer

The virtualization layer provides virtual machines and their basic policy enforcement capabilities. We built on existing versions of the L4 and Xen hypervisors. Our main focus is to extend these hypervisors to increase security. That includes several aspects:

- Fine-grained Trust Domains: Unlike today's version of Xen, we separate services into small isolated virtual machines to increase robustness and security.
- Policy-enforcement: The virtualization layer is built to enforce a wide range of security policies. Examples include access and flow control policies as well as resource sharing policies.
- Verifiable security: By means of trusted computing, external stakeholders can verify the virtualization layer and its policies.

The virtualization layer offers a basic management interface (BMI) to the security services layer. The interface supports functions like creating a virtual machine while specifying its virtual network cards, memory, storage, and CPUs. An example of a policy that can be enforced at the virtualization layer are sHype policies that can be loaded at boot-time [SJV+05].

## 4.2 Security Services Layer

The security services layer provides scalable security and virtualization functions that are needed to enforce security policies. This includes *compartment security management*, *user security management*, and *secure device virtualization*.

The compartment security manager manages the life-cycle and tracks the security policies and other context associated with each compartment. This includes integrity constraints, permissions, and global identifiers for each compartment. The compartment security manager can be used to prove selected security properties to peers. The user security manager manages the users of the system and enables authentication of individual users and their associated roles.

An important contribution to scalability for trusted computing is the focus on security properties for trust management [PSHW04, SS05, HCF04]. Instead of verifying integrity by means of cryptographic checksums, we use higher-level properties such as user roles, machine types, or trust domains to determine trust. This is done by first using checksums to verify the core security services and then use these security services to evaluate the desired security properties. Only if these properties are satisfied, certain actions such as unsealing a key or performing a transaction with a peer are performed. The consequence is that a verifier only needs to define security properties to be satisfied and no longer needs to track individual software configurations that are deemed trustworthy.

Virtualized devices can include any device that can be made to support virtualization.Secure storage provide virtual partitions with integrity, confidentiality, and freshness guarantees. Virtual networks can provide mutually isolated virtual network topologies and secure transport (cf. [GJP+05]). The implementation of trusted user interfaces depends on the environment. A simple solution that is sufficient for reliable selecting a compartment can be implemented by a secure hot-key that is caught by a virtualized keyboard driver. Another alternative is a multi-compartment graphical user interface that assigns a distinguishable window to each compartment. An third option are remote user interfaces such as a secure shell management console or remotely accessible management service. In our secure transaction scenario (described later in Section 5), the user can use a hot-key to switch compartments. In a server setting, the shell will indicate the compartment that it is operating on. The cryptographic services include a virtual TPM as well as other cryptographic and key management primitives.

For efficiency, the security services can push policies into policy enforcement functions of the virtualization layer. This is done, if fast policy enforcement is critical for performance. E.g., a policy decision whether a certain network card can be assigned to a newly created virtual machine can easily be done outside the hypervisor since it is usually not performance critical. Access decision for shared resources, on the other hand, should be executed in the core since their performance is critical.

## 4.3 Virtual Machines Layer

The virtual machines layer contains the actual virtual machines that constitute the payload of the architecture. The architecture can host windows and Linux virtual machines. This is done by providing drivers for accessing the virtual hardware provided by the lower layers. Depending on the hypervisor, certain security services can be implemented by a set of security management machines.

## 4.4 Application Layer

In a management virtual machine, we host the management applications that allow users to interact and maintain their platform. This includes accepting/rejecting policies and defining or loading baseline policies that can delegate certain management functions (such as trust in public keys) to other parties. Another example is the life-cycle management of a trusted platform module (TPM).

An important class of applications are management applications. In particular, in virtualized data-centers, a scalable management infrastructure is essential. Technically, this scalability is achieved by multiple mechanisms such as secure migration of virtual machines that enables load balancing or self-services machines that obtain maintenance orders and execute these orders while only reporting results to the management servers. An example of such a pull model is patch management in which a machine pulls the latest patch policy, then installs the patches from a cluster of software distribution servers, and finally reports its success to the configuration management system. As a consequence, central management infrastructure only manages policies while the costly operations are distributed onto the individual machines.

## 4.5 Implementation of the Architecture

On the L4 hypervisor, the security and management services are isolated virtual machines that run directly on top of the L4 micro kernel. Each service defines a well-defined interface for inter-process communication (IPC). Interaction between services or between instances of hosted payload virtual machines and services is performed by using these interfaces. An IPC call that is issued by a process first goes to the L4 micro kernel, which then transfers it to the callee. The IPC mechanism is implemented similarly to the IPC architecture of CORBA.

The implementation of the security and management services on the Xen Hypervisor is split into two parts. The low-level part is implemented directly in the Xen Kernel running with full privileges. This part contains the security enforcement of the security services. The lower-level part controls the basic access, communication and enforcement and provides a well-defined interface to the higher layers. The higher level includes non-enforcement parts of the security services as well as the management components. Both run in one or more[4] service virtual machines or in a special security service virtual machine as normal user processes.

[4]For increased security, we split the single management virtual machine of Xen into multiple smaller ones.

# 5 Application scenarios

The architecture described in Section 4 is designed to support a wide range of scenarios. To validate our design, we focus on three scenarios with substantially different security requirements that we describe in the sequel.

## 5.1 Private Electronic Transaction (PET)

Private Electronic Transaction (PET) is the first application scenario that we are currently implementing. This simple proof of concept enables users to perform secure SSL transactions. Secure means that (a) a user can validate a given virtual machine that is used for commercial transactions and can convince others of its integrity, and (b) that the user secrets are securely stored throughout the life-cycle of the virtual machine. This scenario illustrates how trusted computing technologies can be used for protecting the end-user while performing security-critical activities using his personal computer.

PET is focused on electronic transactions (e.g. home banking and auctioning) performed with well-known and trusted entities. One compartment will be dedicated for running a trusted browser to be used for the electronic transactions. All other user applications will run in one or more compartments other than the trusted one. The PET application run in the trusted compartment could be provided to the customer as an image on a CD by a trusted third party, delivered and installed on the OpenTC platform in a secure manner. Another possible option would be that the compartment is prepared by other parties (e.g. the OpenTC consortium) and validated by the third party.

A demonstrator for this scenario is currently under development and is expected to be available at the end of this year. Though this demonstrator is focused on the PET scenario, it will include many relevant components of the final OpenTC architecture, e.g., some trusted serviced like the Compartment and the Storage Managers which will be also used in other usage scenarios. Therefore, this demonstrator can be also considered as a preliminary prototype for the other scenarios.

## 5.2 Virtual Data Center (VDC)

Virtual Data Center (VDC) is a scenario that focuses on performance of large scale systems. The core idea is that a service provider runs a data center for multiple customers (usually corporations). These customers require that certain infrastructures with certain security requirements are provided while being isolated from all other customers.

The service provider has a virtual IT infrastructure (consisting of virtual machines, virtual network cards, virtual network connections, virtual firewalls, virtual TPMs, etc.) and the data center infrastructure provider maps the virtual IT infrastructure onto the physical data center infrastructure. In such a scenario, the data center provides 'virtualized' resources rather than real, physical ones and these resources effectively share the underlying physical infrastructure.

The management of such a data center will be split into two layers. The virtual hardware will be managed by our new management applications. This virtual infrastructure management replaces the time consuming manual configuration of systems. The content of the virtual infrastructure can then be maintained using normal systems management tools. Access to infrastructure management facilities will be more constrained and only a small number of well defined options will be available to the service provider. Even the rights of the data center operator will usually be limited in order to prevent cross-customer connections by accidental misconfiguration. Without having such technical means to validate that the provider does not abuse these privilege, the service provider may have to put complete trust on the infrastructure provider. Customers may therefore require attestations at runtime that their services are protected from such unauthorized access. Naturally, such

attestations need to hide all details about other customers on the same physical machine while still providing convincing security proof.

The customer components will finally be hosted in compartments. A compartment is characterized by services and applications it hosts, its configuration and policies attached to it. These policies define the protection level for the accessed data and processed, the protection level of applications and services that participate in the processing of data, the information flow between different compartments (both local to the hardware platform and on remote platforms), the permitted interactions with the virtualization environment and management, the events that trigger a change in the trust state of a compartment and/or its execution environment.

### 5.3 Corporate Computing on Home PC (CCHP)

Corporate Computing on Home PC (CCHP) is a scenario that focuses on multilateral security requirements. While a user wants to freely use his home PC, his/her employer wants to ensure the integrity and confidentiality of corporate data stored on the PC. That requires isolation between the company-specific data/applications and other data/application as well as remote verification of the parts that are security critical for the enterprise by the employer. Our design for this scenario involves establishment of two virtual zones – one for the user and one for the enterprise. Both the home and the enterprise zone can host one or more virtual machines. At a very high-level, isolation involves ensuring that even if one virtual machine is compromised (e.g., a virus in the compartment hosting the user's favorite online game), it should not affect other virtual machines.

The two actors, company and employee, normally have a mutually distrustful (not to be confused with mistrustful) relationship. This means that the corporation does not completely trust a home user to maintain his/her machine correctly. The home PC user needs assurance that the corporation can only see data in the corporate zone while not being able to learn about information and activities in the home zone. In both cases, the underlying principle is "trust but verify." The actors differ in their concerns. The company implements the mechanisms that allow the remote virtual machines (running on employees' home PCs) appropriate access into the corporate network after authentication and verification. The employee owns and provides the trusted platform hosting the corporate virtual machines.

A particular technical challenge in this scenario is protection of corporate secrets during the life cycle of the virtual machines. A home user may install, start, delete, hibernate, or migrate a virtual machine. He may also choose to sell his PC including the corporate virtual machines. In all these cases, corporate data needs to remain properly protected.

## 6 Conclusion

We have described an open and scalable architecture for trusted virtualization. Trusted computing provides a hardware root of trust while virtualization enables incident containment and fine-grained security policy enforcement.

The core technical challenges that we face are multi-lateral security that balances requirements of all stakeholders as well as an open approach to trusted computing that is not limited to particular policies or mechanisms.

A main focus of the project is openness and scalability. Openness is addressed by an open design and open-source implementation which separate security policies from enforcement mechanisms. Scalability must be addressed by all parts of the architecture. Important aspects are scalable attestation, management, and platform. While scalable attestation is provided by property-based attestation that separates trust from integrity checksums of software, scalable management is addressed by enabling each machine to poll for and implement policy updates. This reduces the load on enterprise management infrastructures. The scalability of the platform has many aspects. One aspect is our support for

secure migration of virtual machines. This allows us to subsequently add new hardware to a data center while the data center then automatically migrates virtual machines in order to balance the overall load.

The ongoing implementation of different scenarios will provide further validation and insight into additional challenges when implementing such a secure trusted computing infrastructure.

# Acknowledgments

This article is based on input by many members of the OpenTC project consortium. We would like to thank in particular David Plaquin for the documentation of the Basic Management Interface, Ahmad Sadeghi for comments on related work, and Bernhard Jansen for comments on our architecture survey.

This work has been partially funded by the EC as part of the OpenTC project [Ope] (ref. nr. 027635). It is the work of the authors alone and may not reflect the opinion of the whole project. The OpenTC project includes 23 partners such as HP, IBM, University of Cambridge, Ruhr University of Bochum, and SUSE.

# A PET Threat model and Countermeasures

This appendix describes the threat model which PET is built upon and the technical countermeasures currently under development for defending the client system. These countermeasures are implemented through the components of the OpenTC architecture described in Section 4.

## A.1 Assumptions

For the PET scenario the following assumptions apply.

**Correct hardware** The underlying hardware (e.g. CPU, devices, TPM, etc.) is non-malicious and behaves as specified.

**The TCB is secure and trusted** The Trusted Computing Base of the client platform, including the virtualization engines and the security services is correct and behaves as specified.

**Correct trusted credentials** It is assumed that the server's credentials of the trusted party - all certificates on the certification path for the TLS server authentication, from the root CA to the server certificate, both included - are correct and have been "installed" in a secure manner.

**User credentials** Authentication schemes other than the HTTP basic authentication have not been taken into account in this scenario. Therefore only user name and password are supported: credentials like one time passwords and the TLS client authentication are currently unsupported.

**Trusted Administrator** The standard services for compartment administration and platform management must be trusted to act in accordance with the wishes of users, since they have to access security-critical information.

**No physical attack** Physical attacks against the underlying hardware platform have not been taken into account in this scenario.

**Trusted party** The actual bank or auction house is considered as a trusted party: this means that it is assumed that these parties handle all sensitive data of users securely.

## A.2 Threats

The PET scenario takes into account the following threats.

**Phishing** An attacker tries to impersonate the trusted party's (e.g. bank's) web site. This way, confidential data such as the credentials (user name, password, PINs, TANs) might be disclosed. The attacker can then use this information to illegally withdraw money from the user's bank account. The phishing attack can therefore be considered as a sequence of three different sub-attacks: redirecting the user to a fake web server, getting the user's credentials, using them on behalf of the legitimate user.

**Network redirection to a fake web server** Network redirection to a fake bank server due to pharming, DNS cache poisoning and similar attacks.

**Trojan horse and Malware** Potentially harmful software such as Trojan horses or key loggers might get installed on a computer system as a consequence of the user's actions but without his/her knowledge or approval. This software might report sensitive data of the user to an attacker.

**Exploitation of software vulnerabilities** A remote attacker can try to exploit the software vulnerabilities present in the user's system by running arbitrary code in order to get the control of the platform.

**Modification of the client configuration** The user may intentionally or unintentionally modify the configuration of his/her client platform, also including the PET application.

## A.3 Countermeasures

Under the previous assumptions, a set of technical solutions suitable for defending the user's platform against the listed threats has been defined. In order to find the sets with the minimum number of countermeasures that protect the whole user's system, we have modeled the security threats by using the well-known attack trees [Sch99] formalism. We then extended it into a new formalism, called attack-countermeasure trees and currently under development by attaching a proper countermeasure pattern to each each attack in a leaf.

The resulting trees can be mapped to binary expressions where the leafs of the tree represent binary variables - countermeasure used or not - and the root of the tree represents the result of the expression - attack successful or not. It is then possible to find the minimum sets of countermeasures that satisfy the boolean equation obtained by forcing the boolean expression to be 'false.' Among these minimum sets, we chose two configurations of countermeasures that will be supported by the OpenTC architecture currently under development. The configurations differ primarily in whether modifications are required at the trusted party (e.g., bank), compared to the current manner of doing web transactions. We now describe those countermeasures.

### A.3.1 Countermeasures Shared by Both Configurations

Some countermeasures are shared between the two proposed configurations. They mainly rely upon the isolation property provided by the virtualization engines. The PET application will run on a standard web browser (possibly customized) in a stripped-down dedicated compartment, i.e. a Xen guest domain or a L4Linux instance, considered as "trusted." All applications commonly used, like the e-mail client, run in a different (or in different) compartment(s) considered as "untrusted." The full TCB and the trusted compartment running the browser for the PET application are always measured at their startup before being executed.

The user can switch between the compartments by using two different "hot-key" sequences directly handled by the TCB that implements trusted paths to the user. In the final version of the OpenTC framework, more advanced tools for the management of the compartments will be available. Since the trusted and the untrusted compartment are isolated an application running within the latter is not allowed to launch or switch to the former. Therefore, if the user receives a fake e-mail from a phisher, when he/she clicks on the fake link only the untrusted browser will be opened.

Furthermore, the current demonstrator won't provide any reference monitor for dynamically verifying the integrity status of the compartment and for taking the appropriate actions (e.g. stopping the compartment) should the verification fail. Therefore a possible way for protecting the trusted compartment against the software vulnerabilities exploitation is assuring that the network connections to/from the trusted compartment will occur only with the trusted party. As a partial protection against this threat, a firewall for filtering all incoming network packets will be setup for the trusted compartment.

In both configurations, the standard authentication of the server performed during the end-to-end TLS handshake between the browser and the trusted party web site will be used to trigger other the actions of other countermeasures.

Finally with both configurations, the user won't be able to access the trusted party web site from the untrusted compartment.

### A.3.2 Configuration with Modification at the Trusted Party Side

The first chosen configuration relies upon the mandatory remote attestation of the client performed by the bank server. This configuration's specific countermeasures are the following.

**The trusted compartment and the untrusted one looks very different** The user is clearly notified if he/she is using the trusted compartment or not.

**Browser changes look when it is securely connected to the bank** The appearance of the browser like a skin or a background image will change only when connected to the right web site. This look, bound to the web site, is chosen by the user and it is securely sealed by using the TPM. The change of appearance will be triggered by a successful TLS connection and it will occur only within the trusted compartment.

**Client remote attestation** The web server of the bank or other Trusted Party always requests the client for its attestation before: the client will report the integrity measurement of the whole TCB and of the trusted compartment to the trusted party.

**Always requesting the trusted party's platform authentication** All network traffic outgoing from the trusted compartment is routed to a dedicated compartment - to be considered as part of the TCB and therefore correct - that forwards the network traffic coming from the trusted browser to the Internet only if directed to the right web site. For authenticating the right web site before establishing the end-to-end TLS channel between the

browser and the trusted party, we chose to always require the authentication of the trusted party's platform as a first step toward the full remote attestation.

On the client side, the client remote attestation protocol and the trusted party's platform authentication and will be performed by a proxy running in the dedicate compartment.

### A.3.3 Configuration without Any Modification at the Trusted Party Side

The second chosen configuration requires modifications only at the client side and uses a protected storage for the user's credentials. The specific countermeasures for this configuration are the following.

**Protected storage for the user's credentials** The protected storage works within the trusted compartment only, seals the user's credentials once they are set the first time and releases them only when connected to the right web site; the identification of the right web site is triggered by a successful TLS server authentication.

**Strong validation of the server certificate** This is a companion countermeasure of the standard TLS authentication of the server and it means that in addition to the standard certificate validation of the server certificate (i.e. CA's signature verification and revocation checking) must be done, but also checking that is the "right" instance of the certificate. Furthermore the only CA certificate present in the browser's database of the trusted certificates.

**Checking the compartment's integrity before booting it** The Compartment Manager will securely store the measurement of the trusted compartment and it will check it at every startup against the right measure. If the measures differ from those stored by the Compartment Manater, the trusted compartment won't be started.

# References


[BDF+03] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[CJPL02] A. Carroll, M. Juarez, J. Polk, and T. Leininger. Palladium: A business overview, August 2002. `http://www.microsoft.com/PressPass/features/2002/jul02/0724palladiumwp.asp`.

[GJP+05] J.L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. Van Doorn, and R. Caceres. Trusted Virtual Domains: Toward secure distributed services. In *Proc. of the First Workshop on Hot Topics in System Dependability (Hotdep05)*, Yokohama, Japan, June 2005. IEEE Press.

[GPC+03] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: a virtual machine-based platform for trusted computing. In *ACM Symposium on Operating Systems Principles (ASOSP)*, pages 193–206. ACM Press, 2003.

[GR05] Tal Garfinkel and Mendel Rosenblum. When virtual is harder than real: Security challenges in virtual machine based computing environments. In *Proceedings of the 10th Workshop on Hot Topics in Operating Systems (HotOS-X)*, May 2005.

[Groa] Applied Data Security Group. Trusted GRUB. http://www.prosec.rub.de/trusted_grub.html.

[Grob] Operating Systems Group. Fiasco micro-kernel. http://os.inf.tu-dresden.de/fiasco/.

[HCF04] Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic Remote Attestation - virtual machine directed approach to Trusted Computing. In *Virtual Machine Research and Technology Symposium*, pages 29–41, 2004.

[HHL+97] H. Härtig, M. Hohmuth, J. Liedtke, S. Schönberg, and J. Wolter. Performance of $\mu$-kernel-based systems. In *16th ACM Symposium on Operating System Principles (SOSP)*, St. Malo, France, October 1997. ACM Press.

[Hoh98] M. Hohmuth. The Fiasco kernel: Requirements definition, 1998.

[Mica] Bitlocker Drive Encryption . http://www.microsoft.com/technet/windowsvista/security/bitlockr.mspx.

[Micb] Microsoft Windows Vista. http://www.microsoft.com/Windowsvista/.

[MN00] R. Meushaw and D. Simard. NetTop. NetTop: Commercial technology in high assurance applications, 2000. http://www.vmware.com/pdf/TechTrendNotes.pdf.

[MSMW03] Rich MacDonald, Sean Smith, John Marchesini, and Omen Wild. Bear: An open-source virtual secure coprocessor based on TCPA. Technical Report TR2003-471, Dartmouth College, August 2003.

[MSW+04] J. Marchesini, S.W. Smith, O. Wild, Josh Stabiner, and A. Barsamian. Open-source applications of TCPA hardware. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 294–303, 6-10 Dec. 2004.

[MSWM03] John Marchesini, Sean Smith, Omen Wild, and Rich MacDonald. Experimenting with TCPA/TCG hardware, or: How i learned to stop worrying and love the bear. Technical Report TR2003-476, Dartmouth College, December 2003.

[NSA] NSA. Security-Enhanced Linux. http://www.nsa.gov/selinux/.

[Ope] Open Trusted Computing (OpenTC). http://www.opentc.net/.

[PSHW04] Jonathan Poritz, Matthias Schunter, Els Van Herreweghen, and Michael Waidner. Property attestation — scalable and privacy-friendly security assessment of peer computers. Technical Report RZ 3548 (# 99559), IBM Research Division, 05/10/2004 2004.

[Sch99] Bruce Schneier. Attack trees: modeling security threats. *Dr. Dobb's Journal* , 1999.

[SJV+05] Reiner Sailer, Trent Jaeger, Enriquillo Valdez, Ronald Perez, Stefan Berger, John Linwood Griffin, and Leendert van Doorn. Building a MAC-based security architecture for the Xen open-source hypervisor. Research Report RC23629, IBM Research Division, June 2005.

[SS05] Ahmad-Reza Sadeghi and Christian Stüble. Property-based attestation for computing platforms: caring about properties, not mechanisms. In *NSPW '04: Proceedings of the 2004 workshop on New security paradigms*, pages 67–77, New York, NY, USA, 2005. ACM Press.

[SvW04] Reiner Sailer, Leendert van Doorn, and James Ward. The role of TPM in enterprise security. Technical Report RC23363, IBM Research Division, October 2004.

[SZ03] David Safford and Mimi Zohar. A Trusted Linux Client (TLC). Technical report, IBM Research Division, 2003.

[vmwa] VMware ESX Server: Platform for virtualizing servers, storage and networking. http://www.vmware.com/pdf/esx_datasheet.pdf.

[vmwb] VMware ESX support to Intel's VT technology . http://www.vmware.com/news/releases/intel_virtualization.html.