Design and Implementation of a Portable ID Management Framework for a Secure Virtual Machine Monitor

Manabu Hirano1,2, Takeshi Okuda2, Eiji Kawai2 and Suguru Yamaguchi2

1 Department of Computer and Information Engineering, Toyota National College of Technology, Aichi, 471-8525, Japan
hirano@toyota-ct.ac.jp

2 Graduate School of Information Science, Nara Institute of Science and Technology, Nara, 630-0192, Japan
{okuda, eiji-ka, suguru}@is.naist.jp

Abstract: A commonly used virtual machine monitor (VMM) allows multiple operating systems to share physical hardware resources as virtual resources in a safe manner. It provides a strong isolation mechanism between virtual machines (VMs). In this paper, we state the importance of ID management for a security-purpose VMM system to enforce security policy on an end-user environment. We present a design of a portable ID management framework for a security-purpose VMM. Our proposal employs a smart card (ID card) for user authentication. The proposed ID management framework can provide generic programming interfaces to existing VMM software. Our ID management framework realizes an authentication between a VMM and its users, an authorization for a VM boot operation, storage of cryptographic keys for VMM-layer’s disk encryption/decryption, and access control for virtual/physical resources based on a user identity. In this paper, we show the prototype implementation of our ID management framework and its integration into the proven VMM software, QEMU.

Keywords: ID Management, Authentication, Virtual Machine Monitor, VMM, Hypervisor, Smart card, ID card

1. Introduction

A virtual machine monitor (VMM), also called a hypervisor, allows multiple operating systems to share physical hardware resources as virtual resources in a safe manner. An ideal VMM technology provides complete isolation of virtual machines (VMs) [1]. Especially, a layer of a trusted VMM can, without modifying a guest operating system, provide useful and strong security functions transparently for each VM, such as storage encryption, traffic confidentiality using a virtual private network (VPN) and other access control mechanisms for physical resources. We call this kind of VMM system “a secure VMM”. We assume that a secure VMM system for an end-user environment is used as a foundation of security policy enforcement in an organization. In recent research, some security-purpose VMM systems have been developed. However, these systems do not consider ID management on the VMM layer. In this paper, we show an ID management framework for these security-purpose VMM systems.

At first, in section 2, we summarize related work on security-purpose VMM systems such as sHype, NetTop, Terra and their security mechanism. In section 3, we show a model for a typical secure VMM system and a problem we have found with it. Section 4 shows the design of our portable ID management framework for a secure VMM system. In section 5, we present our prototype implementation using a smart card and middleware systems. We describe our future work in section 6. We conclude with a summary in section 7.

2. Related Work on a Secure VMM system

A VMM is a technology to encapsulate an operating system, a technology which was originally designed and developed for mainframe computers such as IBM VM/370 [2]. It was intended to support legacy software in a new hardware environment. In recent research, some VMM systems designed for security-purposes have been developed. In this section, we summarize the related work on secure VMM systems.

sHype. sHype (Secure Hypervisor) [3][4] is developed by IBM Research. The subject of the sHype project is to provide a secure foundation for server platforms. sHype provides an access control and an isolation mechanism of virtual resources in hypervisor software. sHype is intended to provide a mandatory access control (MAC) for inter-VM communication in VM coalitions. sHype is currently implemented on the Xen hypervisor [5], rHype (Research Hypervisor) [6], etc. sHype implementation supports some major security policy mechanisms such as the Chinese Wall policy and the simple Type Enforcement policy. sHype mainly consists of: an access control module for virtual resources and inter-VM communication, hypervisor mediation hooks, callback functions and a policy management VM. sHype also supports a TPM (Trusted Platform Module)-based attestation mechanism [7][8].

NetTop. NetTop [9] is designed by NSA. NetTop is constructed with commercial off-the-shelf technology, the VMware product [10], which started in DARPA-sponsored research at Stanford University, and the SELinux as a host operating system. NetTop provides security mechanisms based on isolation of VMs, reliability of a host OS and VMM
software. NetTop project also states the importance of trusted BIOS because all host platform security is dependent on an initial boot-up process. NetTop mainly provides a transparent VPN VM and a Filtering VM that work with an end-user OS like Microsoft Windows. NetTop is intended to be deployed in a governmental environment. NetTop also provides a secure data transfer mechanism, called the “Regrade” server protocol, between two VMs with different security levels. The regrade server achieves data transfer based on a token-based user identity and a regrade policy. It also checks and sanitizes malicious content, and records audit logs. NetTop also provides transparent storage encryption and coalition support.

Terra. Terra architecture provides a trusted virtual machine monitor (TVMM) that isolates and protects independent virtual machines [11]. Terra architecture is designed to build a trusted computing platform using VM technology. Terra architecture supports a remote attestation mechanism to establish a trusted path. A trusted path is achieved by attestation certificate chains for BIOS, a boot-loader and a VM image. Terra also supports an encrypted disk or an integrity-checked disk in the VMM layer. A prototype implementation of Terra architecture employs The VMware ESX server product and some python scripts.

2.1 Security Mechanisms
Most security frameworks using VMM technology are designed carefully to maintain performance and security. Therefore, it is preferable that security codes in a VMM are as small and simple as possible. In addition, security frameworks described above are dependent on the security and reliability of the VMM layer, the underlying BIOS, and a boot-loader. So security frameworks based on VMM technology should guarantee tamper-proof BIOS firmware, a boot-loader and secure VMM codes. For examples, sHype supports TPM-based attestation to check the integrity of hypervisor software and VM images. NetTop and Terra also state the importance of a trusted BIOS and a reliable execution mechanism for BIOS, a boot-loader, VMM software and VM images. Terra states the importance of a trusted path by a remote attestation mechanism. A secure VMM should support a secure and reliable bootstrap mechanism [12].

Security frameworks using VMM technology are based on the isolation mechanism of that technology. Each VM on same VMM can not influence the other VMs. For this mechanism, we can separate the purpose of each isolated VM. For example, one VM can connect to the Internet, but another protected VM can connect to an intranet only. NetTop takes this approach to reduce the number of physical machines for different purposes and different security levels in a governmental environment.

NetTop and Terra support a transparent storage encryption mechanism. The advantage of this mechanism is that it does not require modifying the guest operating system. The secure VMM layer can provide storage encryption and decryption automatically. Even a guest operating system is attacked; an encryption key is not leaked – on the assumption that a secure VMM layer is strongly protected from untrusted guest operating systems and other malicious actions. NetTop also provides a transparent VPN function for guest operating systems using a VPN gateway VM. This approach achieves simple and practical security. It has the same effect as two physical separated machines, an end-user machine and an administrator-controlled VPN gateway server.

We summarize the security mechanism of a secure VMM system as follows: (1) simple design and minimum overhead for security processing with high performance and high reliability; (2) trusted BIOS, trusted bootstrap architecture and an attestation mechanism for core components (a boot-loader, VMM software and VM images). So we need a software measurement mechanism such as TPM to check software integrity; (3) a strong isolation mechanism for each VM; (4) a mandatory access control (MAC) function based on an access control list (ACL) or a security policy (e.g. the Chinese Wall policy, the Type Enforcement policy). The MAC function basically consists of a policy decision point, a policy enforcement point (i.e. hooks for virtual/physical resources), and a policy management function to update and validate distributed policy files; (5) transparent security functions such as automatic storage encryption and traffic confidentiality by a VPN. These security functions will provide useful components for a secure VMM system.

3. Practical Model for a Typical Secure VMM System

Fig.1 depicts a model for a typical secure VMM system described in section 2.1. A key component of a secure VMM system is mandatory access control (MAC) based on a trusted VMM layer. A VMM layer controls the virtual/physical resources from each VM. In this paper, the word resource refers to devices such as a physical NIC which handles network packets, mass storage, a USB device like a thumb drive which holds user data, and other peripherals such as a printer, and so on. A Policy Decision Point (PDP) in Fig.1 processes an authorization decision based on VM requests for virtual/physical resources and VM identity (VM #1, #2 or #3). Policy management updates and validates a security policy.
which is distributed by a central policy server in an organization. A hook point or a Policy Enforcement Point (PEP) are used to enforce the security policy on VM requests. PEP interacts with PDP to retrieve an authorization result.

Figure 2. Access control models for a secure VMM system: (A) is VM ID-based access control, (B) is access control based on both a USER ID and a VM ID.

Transparent security functions add useful and practical security functions to a secure VMM (storage encryption/decryption, VPN service, etc.).

To check reliability of a secure VMM platform, an attestation mechanism validates boot-loader software, secure VMM execution codes, and so on. The bootstrap process should also be executed securely. Attestation and trusted bootstrap are key technologies for a secure VMM system.

A secure VMM (Fig.1) provides a MAC function. The foundation of a MAC function is dependent on there being only one administrator that can access a secure VMM system. Therefore, an end-user cannot bypass the administrator-controlled secure VMM layer. Thus, a secure VMM can provide a policy enforcement mechanism for an end-user environment.

3.1 Problem Statement

In existing secure VMM systems, a VM entity is identified by VMM software, and a secure VMM enforces a security policy based on each VM identity (VM ID). However, to deploy a secure VMM for a practical operational environment, we need to manage a user-identity in a secure VMM. Fig.2 shows access control models for a secure VMM system. (A) in Fig.2 shows access control based on a VM ID only. (B) in Fig.2 shows access control based on both a VM ID and a user identity (USER ID). In this case, secure VMM software authenticates a user. Therefore, secure VMM software can manage a USER ID, and can apply the USER ID to the access control mechanism. In a conventional system ((A) in Fig.2), secure VMM software does not authenticate a user; a user authentication mechanism is normally dependent on a guest operating system (e.g. Windows Logon, Unix user/password). A conventional system cannot handle the mapping between a VM ID and a USER ID in a VMM layer.

From the perspective of a security administrator, ID management on multiple guest operating systems is not efficient, because the administrator have to maintain multiple user IDs on multiple guest operating systems. Our proposal assumes that a trusted VMM layer can provide many useful security functions transparently to administrators. Introducing user identification and authentication in a secure VMM system has many advantages. First, a secure VMM will be able to control many VMM functions based on a USER ID, such as a VM boot operation and disk encryption. Furthermore, access control for virtual/physical resources will be more effective using an authenticated USER ID because a security administrator in most organizations prefers a USER ID-based authorization, instead of a VM ID-based authorization only. The USER ID-based authorization also enables a secure VMM to record audit logs based on a USER ID. An audit log is an essential component of security systems. An audit log also achieves non-repudiation of user actions on an administrator-controlled secure VMM system. In this paper, we describe a user authentication and an ID management framework using a smart card (ID card) for a secure VMM system.

4. 4 Design of a Portable ID Management Framework for a Secure VMM system

4.1 User Authentication for End-point Security Enforcement

Fig.3 shows the basic design of the portable ID management framework for a secure VMM. This example employs a user identity to control a VE boot process. We assume that each employees of an organization has their ID card, and that the ID card is a multi-application smart card. Our proposed smart card stores a Secure VMM application which supports PKCS#11 compatible functions. So our proposed ID management framework can be used with many smart card products. A secure VMM ID management API in a secure VMM is a high level API using PKCS#11 middleware and underlying libraries such as a PC/SC (Personal Computer/Smart Card) library and a device driver of a smart card reader.

A secure VMM ID management API provides user authentication functions based on Public Key Infrastructure (PKI) for secure VMM systems. Our proposal also provides a standard ID/password authentication API without a smart
card. We mainly employ PKI technology to manage an employee’s identity of an organization. Governments in Belgium, France and many countries are considering a national ID card system based on PKI to authenticate a citizen’s identity. ID management based on PKI technology is growing in the public sector world wide.

![Diagram](image1)

**Figure 4.** Sequence of user authentication: A user logs in to an administrator-controlled secure VMM system.

![Diagram](image2)

**Figure 5.** Example usage of the ID management framework (Transparent VM image encryption): A smart card (ID card) can store a user-specific key (e.g. a 256 bit AES key) for storage encryption or for other user-specific binary data.

Fig. 4 shows the details of user authentication in a secure VMM system. This process is a simple challenge and response authentication procedure. A user inputs a PIN number, and a smart card authenticates the user. An authentication handshake is executed between the smart card and an ID management system of the secure VMM. The secure VMM sends a challenge as R, and the smart card then returns the user’s Public Key Certificate (PKCUSER) and a signature of R (RUSER) generated by the user’s private key (PrivateKeyUSER). The ID management system validates the certificate chain using a trust anchor certificate for a VMM (PKCTrustAnchorVMM). In addition, an ID management system checks certificate revocation status by a Certificate Revocation List (CRL) or some other mechanism like an Online Certificate Status Protocol (OCSP). A secure VMM and the ID management software should be protected from malicious attacks by an attestation mechanism and an isolation mechanism of VMM. If authentication is successful, then a secure VMM will be able to enforce security policy based on a USER ID in the protected VMM layer. Thus, an administrator-controlled secure VMM and the ID management framework provide the fundamental part of an end-point policy enforcement mechanism in an organization.

<table>
<thead>
<tr>
<th>Table 1. Hardware and software environment for prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Smart card</strong></td>
</tr>
<tr>
<td><strong>Smart card reader</strong></td>
</tr>
<tr>
<td><strong>OS</strong></td>
</tr>
<tr>
<td><strong>Cryptographic libraries</strong></td>
</tr>
<tr>
<td><strong>PC/SC library</strong></td>
</tr>
<tr>
<td><strong>USB CCID driver</strong></td>
</tr>
</tbody>
</table>

Fig. 4 shows unilateral authentication. However, our proposal can provide mutual authentication using a user’s trust anchor certificate (PKCTrustAnchorUSER) in the smart card. A secure VMM system may have a transparent traffic security function like a VPN (VM’s outbound traffic is automatically encrypted and its integrity is automatically checked). The ID management framework can provide mutual authentication for the VPN services by PKCUSER and PKCTrustAnchorUSER.

### 4.2 Example: Transparent VM image Encryption and Decryption

Fig 5 shows an example of encryption/decryption for VM image files. In this case, an ID management framework handles a user’s disk encryption key. An encryption key (EncryptionKeyUSER) is stored in the user’s smart card. A secure VMM decrypts the VM images by retrieving the key from an authenticated smart card. Each encryption key is associated with the user’s identity. So an attacker without a valid and user-specific smart card can not decrypt the VM image. The encryption/decryption of a VM image file is automatically and transparently executed by a hook point in a secure VMM layer. This is a good example of a transparent security function in a VMM layer using our ID management framework.

### 5. Prototype Implementation

We have implemented the portable ID management framework for a secure VMM system shown in Fig.3. In our prototype implementation, we used open source software to develop the portable ID management software. Table 1 shows the hardware and software environment for prototype implementation. We have implemented the ID management software on both FreeBSD and Linux operating systems on x86 architecture. We employed the eLWISE smart card manufactured by NTT Communications. The eLWISE is also used as the Japanese national ID card (i.e. the Basic Resident Register system in Japan). We have implemented original PKCS#11 middleware and its high level API and our portable secure VMM ID management API. We have merged our
portable ID management software with proven open source VMM software, QEMU [13]. In the current implementation, we have realized APIs for VM boot management based on a USER ID and VM image encryption/decryption by a user’s encryption key (Fig.3 and Fig.5). Our portable ID management software is an independent module, so it will be applicable to other VMM software.

6. Future Work

Our prototype implementation is constructed on standard middleware APIs (PKCS#11 and PC/SC), so our software is portable and can utilize many consumer off-the-shelf smart card products. We are currently refining our prototype implementation to reduce some library and system call dependencies, so as to integrate our software into thin and lightweight VMM software which runs directly on hardware (i.e. Type I VMM). We are working towards improved ID management software to achieve OS and library independence, and more portable and higher performance.

In this paper, we show a design and implementation of a secure VMM component only. However, for the operational phase, we need to consider the following systems: an issuance server for an ID card, a revocation management server for a PKC (CRL, OCSP or some other mechanism), a security policy distribution server, and an update mechanism for the ID card’s contents. If a user stores his or her disk encryption key, we must consider a key escrow and recovery mechanism for a lost or stolen ID card. Our framework has a digital signature API for authentication purposes. If a digital signature function is used to guarantee validation and non-repudiation of data, we will need a mechanism to manage key and certificate history.

7. Conclusion

Conventional VMM software does not consider ID management and its standard interface. Our ID management framework defines standard APIs for secure VMM software, such as a smart card-based user authentication API based on PKI, a simple ID/password authentication API without a smart card, a VM boot management API, a storage API for an encryption key, and so on. We have implemented the ID management framework using a standard PKCS#11 compatible smart card and open source software. We are currently refining the ID management software to reduce library and system call dependencies, so as to integrate our software into thin and lightweight VMM software which runs directly on hardware. Our ID management framework can be used to construct a security enforcement mechanism on end-user terminals.

Acknowledgment

This work was supported by Special Coordination Funds for Promoting Science and Technology of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References


Author Biographies

Manabu Hirano was born in Hokkaido, Japan. He received the M.E. degree in computer science from Nara Institute of Science and Technology (NAIST), Japan, in 2002. He worked for TOSHIBA Corporation since 2002 and worked on a mobile Internet-related business venture. Since 2004 he has worked for Toyota National College of Technology, Japan, as a research associate. He is also now a Ph.D. candidate in NAIST. His research interests include all aspects of ID management and its real-world applications.
Takeshi Okuda  received the M.E. degree in information science from Osaka University in 1998. He is currently an assistant professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include network security, mobile agent technology and multimedia application. He is a member of the IEEE.

Eiji Kawai is a research associate professor with Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). He received the B.S. degree in mathematics from Kyoto University in 1996 and the M.E. and D.E. degrees in information systems from Nara Institute of Science and Technology in 1998 and 2001, respectively. He was an awarded researcher with PRESTO, Japan Science and Technology Agency (JST) from 2000 to 2003. His research interests cover various topics relevant to distributed computing environments, such as network protocols, operating systems, security, programming languages, server system design, and performance evaluation.

Suguru Yamaguchi  received the M.E. and D.E. degrees in computer science from Osaka University, Japan, in 1988 and 1991, respectively. From 1990 to 1992, he was an Assistant Professor in Education Center for Information Processing, Osaka University. From 1992 to 1993, he was with Information Technology Center, Nara Institute of Science and Technology (NAIST), Japan, as an Associate Professor. Since 1993, he has been with Graduate School of Information Science, NAIST, where he is now a Professor. Since April 2004, he has been an Advisor on Information Security, at Cabinet Secretariat, Government of Japan. His research interests include technologies for information sharing, multimedia communication over high speed communication channels, network security and network management for the Internet.