# Portable ID Management Framework for Security Enhancement of Virtual Machine Monitors

Manabu Hirano[1)], Takeshi Okuda[2)], Eiji Kawai[9)],
Takahiro Shinagawa[3)], Hideki Eiraku[3)], Kouichi Tanimoto[3)],
Shoichi Hasegawa[3)], Takashi Horie[3)], Seiji Mune[3)], Kazumasa Omote[4)],
Kenichi Kourai[5)], Yoshihiro Oyama[6)], Kenji Kono[7)], Shigeru Chiba[8)],
Yasushi Shinjo[3)], Kazuhiko Kato[3)], and Suguru Yamaguchi[2)]
*[1)] Toyota National College of Technology,*
*[2)] Nara Institute of Science and Technology (NAIST),*
*[3)] University of Tsukuba, [4)] Japan Institute of Science and Technology (JAIST),*
*[5)] Kyusyu Institute of Technology, [6)] The University of Electro-communications,*
*[7)] Keio University, [8)] Tokyo Institute of Technology,*
*[9)] National Institute of Information and Communications Technology*
*Japan*

## 1. Introduction

Most governmental and commercial organizations are processing a massive amount of data everyday. In such organizations, end-users' computers are physically distributed and they can be moved outside of organizations. Therefore, it is difficult to monitor, and enforce security policies to these distributed end-users' computers. Although a security administrator in organizations can install monitoring software, an end-user can remove and bypass this kind of software on conventional operating systems. Most existing commercial operating systems cannot enforce a mandatory security policy to end-users. This chapter shows a security policy enforcement mechanism based on a virtual machine monitor (VMM).

A VMM is a technology to encapsulate an operating system, which was originally developed for mainframe computers like IBM VM/370 (Seawright & MacKinnon, 1979). Original VMM mechanism was intended to support legacy software of mainframe computers in new hardware. An ideal VMM technology provides complete isolation of virtual machines (VMs) (Madnick & Donovan, 1973). This feature enables us to separate domains with different security levels in a same physical machine. A layer of a VMM can, without modifying a guest operating system, provide useful and strong security functions transparently for client and server computers. We call this kind of VMM system ``a secure VMM''. A concept of a secure VMM is useful to insert a security layer without modifying existing operating systems. In this chapter, we use terms, both *VMM* and *hypervisor*, to

express this mediation software. In particular, a *hypervisor* means a bare-metal VMM which does not need a host operating system.

In recent research, some VMM systems designed for security-purposes have been proposed as follows.

**sHype.** sHype (Secure Hypervisor) (Sailer et al., 2005) is developed by IBM Research. The purpose of the sHype project is to construct 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 of server VM coalitions. sHype is implemented on Xen hypervisor and rHype (Research Hypervisor). sHype implementation supports some major security policy mechanisms such as Chinese wall policy and the simple Type Enforcement (TE) policy. Main components of sHype are an access control module for virtual resources and inter-VM communication, hypervisor mediation hooks, callback functions and a policy management VM.

**NetTop.** NetTop is designed by NSA (Meushaw & Simard, 2000). NetTop is constructed with commercial off-the-shelf technology, VMware products and SELinux as a host operating system. NetTop provides security mechanisms using isolation of VMs, and it is based on reliability of a host operating system 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 can provide a transparent VPN VM and a Filtering VM to protect and filter communication channels of end-users' 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 can also check and sanitize malicious content, and record audit logs. NetTop also provides a transparent storage encryption service.

**Terra.** Terra architecture provides a trusted virtual machine monitor (TVMM) that isolates and protects each virtual machine (Garfinkel et al., 2003). Terra architecture is designed to construct a trusted computing platform based on a VMM technology. Terra architecture supports a remote attestation mechanism to establish a trusted path. The trusted path is achieved by attestation certificate chains for BIOS, a boot loader and a VM image. Terra also supports an encrypted disk and an integrity-checked disk in the VMM layer. A prototype implementation of Terra architecture employs the VMware ESX server product and python scripts.

**BitVisor.** We are developing novel secure VMM software called BitVisor (Shinagawa et al., 2009). The first version of BitVisor has been released to the public in March 2008. BitVisor has been developed by 6 universities and college, companies with the help of NISC (National Information Security Center), Japan. BitVisor can be downloaded as complete source codes. Developers can download and extend source codes of this novel security software. BitVisor provides transparent security functions like a built-in IPsec-VPN module with IKEv1 (RFC4301 and RFC2409) and a storage encryption module based on XTS-AES algorithm (IEEE1619 standard) in the VMM layer. The IPsec-VPN module enables users to establish VPN without modifying guest operating systems. The IPsec-VPN module provides

encrypted communication and mutual authentication functions between distributed end-point computers and a VPN gateway. The transparent storage encryption function can force the use of encrypted storages (ex. ATA hard disk drives and USB thumb drives) in end-point computers to prevent information leak cases.

As described above, there are many secure VMM systems. This chapter presents a portable ID management framework for secure VMM systems. The proposed ID management framework provides useful services to VMM software. For instance, the proposed portable ID management framework provides a user authentication function using PKI-based ID cards and a user ID based authorization function for transparent security services in a VMM layer.

In the following sections, we first describe a basic concept of a secure VMM. Then, we show a design and a prototype implementation of the proposed portable ID management framework for secure VMM systems. Then, we present our novel secure VMM software called BitVisor and its integration with the proposed portable ID management framework.

## 2. Overview of Secure VMM Systems

Security frameworks using a VMM technology are based on an isolation mechanism of that technology. Each VM works on same VMM cannot 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 VM can restrict network connections except for intranet accesses. NetTop takes this approach to reduce the number of physical machines for different purposes and different security levels in a governmental environment. We can construct isolated domains with different security levels as different VMs using the VMM technology.

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. It has the same effect as two physical separated machines, an end-user machine and an administrator-controlled VPN gateway server machine.

Figure 1 illustrates a basic concept of a secure VMM architecture. Secure VMM systems have the following features: (1) thin, light-weight, and minimum overhead for security processing with high performance and high testability; (2) a trusted bootstrap architecture and an attestation mechanism for each component (e.g. BIOS codes, a boot-loader, VMM software and VM images). We need a measured launch mechanism using Intel's TXT (Trusted Execution Technology) hardware, formerly known as LaGrande, and TPM (Trusted Platform Module) chip to check software integrity at the boot time; (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 and the Type

Enforcement policy). The MAC function basically consists of a policy decision point (PDP), a policy enforcement point (PEP; i.e. hook points for virtual/physical resources), and a policy management function to update distributed policy files; (5) transparent security functions like automatic storage encryption and encrypted communication channel using a VPN function; and (6) a protection mechanism against run-time attacks to VMM software.

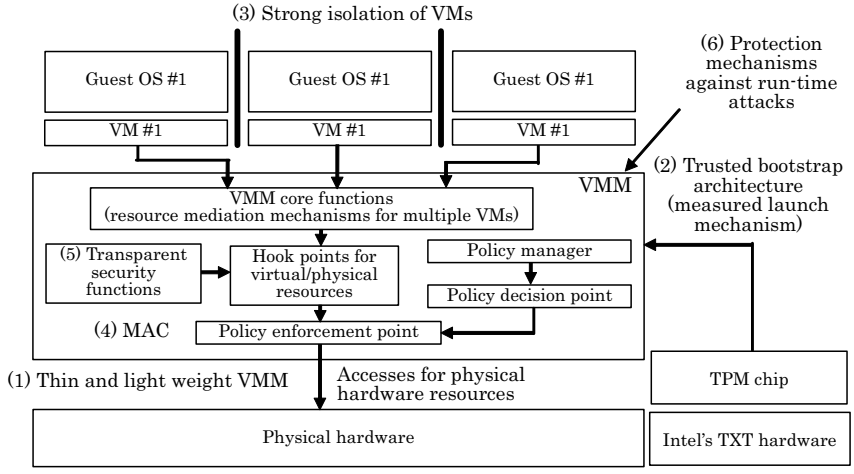


Fig. 1. Basic concept of a secure VMM architecture

A key component of secure VMM systems is MAC functions. The secure VMM software can control virtual/physical resources of each VM. In this chapter, the word *resource* refers to devices like a NIC which handles network packets and a USB thumb drive which holds a user's data. As shown in Figure 1, secure VMM systems hooks I/O data of these devices and they enforce security policies in the PEP. The policy manager updates and validates a security policy which is distributed by a central policy distribution server in an organization. The PDP processes an authorization decision for VM operations and sends the result to the PEP. Finally, the PEP enforces security policies based on retrieved authorization result.

The foundation of a MAC function is dependent on there being only one administrator that can access a management function of a secure VMM system. We have to prevent that an end-user cannot bypass the administrator-controlled secure VMM layer. Secure VMM systems provides a policy enforcement mechanism for end-users' computers based on above methods.

## 3. ID Management Problems in a VMM Layer

In existing secure VMM systems, a VM entity is identified by VMM software, and a secure VMM enforces a security policy based on the each VM identity (VM ID). However, to deploy a secure VMM system as a policy enforcement mechanism, we need to manage a user-identity in a secure VMM system. Figure 2 shows access control models of a secure VMM system. The figure (A) shows an access control model based on a VM ID only. The figure (B) shows an access control model based on both a VM ID and a user identity (User

ID). In the latter case, secure VMM software first authenticates a user and identified her or his user ID. Then, secure VMM software enforces a security policy using the user ID.

In the conventional system like the figure (A), a user authentication mechanism is still dependent on a guest operating system (e.g. Windows Logon and UNIX-like OS's password mechanism). Conventional secure VMM systems cannot handle the mapping between a VM ID and a user ID of a guest OS. From the usability and the security, ID management mechanisms on multiple guest operating systems on multiple VMs are not efficient, because the administrator have to maintain multiple user IDs on multiple guest operating systems on each VM.

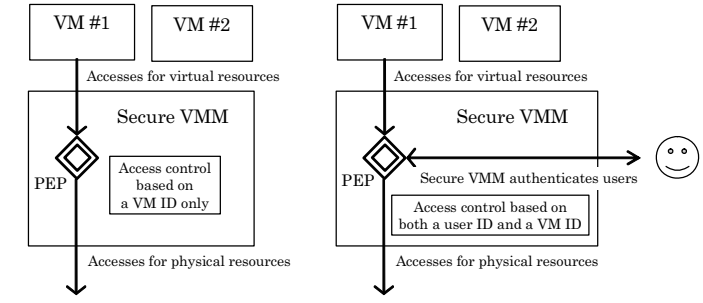


Fig. 2. Access control models of a secure VMM system

Secure VMM layer can provide many security functions transparently to upper guest operating systems. Employing user identification and authentication in a VMM layer has many advantages. First, a secure VMM will be able to control many VMM functions based on a user ID, like a VM boot operation and disk encryption. Furthermore, access control of virtual/physical devices will be 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 the user ID. An audit log is an essential component of security systems. Audit logs also achieve non-repudiation of users' actions on an administrator-controlled secure VMM system. In the following sections, we describe a method to integrate a proposed portable ID management framework based on a PKI-based smart card (ID card) into secure VMM systems.

## 4. Portable ID Management Framework for Secure VMM Systems

Figure 3 shows a design of the proposed portable ID management framework for secure VMM systems. We assume that each employees of an organization has their PKI-based ID cards. We mainly employ PKI technology to manage an employee's identity of an organization. For instance, governments in Belgium, France and many countries are considering a national ID card system based on PKI to authenticate citizens' identities. ID management based on a PKI technology is growing in the public sector world wide. We

show a method to integrate PKI-based authentication and authorization mechanisms into secure VMM systems to enforce security policies for distributed end-users' computers.

Our proposed smart card stores the secure VMM application we developed. Middleware layers in a secure VMM system support PKCS#11 APIs, PC/SC (Personal Computer/Smart Card) APIs, and the minimum cryptographic APIs. The proposed portable ID management framework libraries also include a USB CCID (Chip/smart Card Interface Devices) device driver for generic smart card readers. The secure VMM ID management library provides user authentication functions based on PKI-based ID card and authorization functions based on a security policy. The proposed ID management framework libraries also provide a standard ID/password authentication API. These functions can be used to authorize VM operations based on an authenticated user ID.

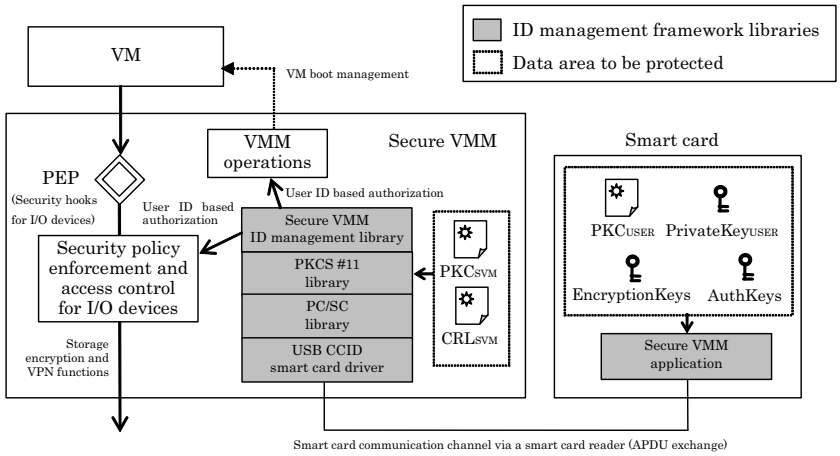


Fig. 3. Design of the portable ID management framework for secure VMM systems

In addition, the secure VMM software can enforce storage encryption/decryption services to guest operating systems transparently. An encryption key ($EncryptionKey_{USER}$) is stored in the user's smart card. Secure VMM software encrypts and decrypts storages using the encryption keys. Each encryption key is associated with the user's identity. Therefore, an attacker without a valid smart card cannot decrypt a user's storage. Secure VMM software also provides a VPN service in hook points of a physical NIC device. PKI-based authentication functions can be used for the initial user authentication between a user's machine and a VPN gateway machine.

Figure 4 shows the details of user authentication of the proposed ID management framework. The user authentication function employs a simple challenge and response mechanism. First, a user inputs a PIN number of her or his smart card, and the smart card authenticates the user. Then, authentication data are exchanged between the smart card and the ID management framework libraries in secure VMM software. Secure VMM software sends a challenge as R, and the smart card then returns the user's Public Key Certificate ($PKC_{USER}$) and the signature of R ($\{R\}_{USER}$) generated by the user's private key ($PrivateKey_{USER}$). The ID management framework libraries validate retrieved $PKC_{USER}$ using

a trust anchor certificate for a secure VMM ($PKC_{SVM}$). The ID management framework libraries check certificate revocation status by a Certificate Revocation List ($CRL_{SVM}$). If the authentication is successful, then a secure VMM will be able to enforce security policy based on a user ID in the secure VMM layer. As a result, administrator-controlled secure VMM software with the ID management framework libraries provides the fundamental part of an end-point policy enforcement mechanism in an organization.

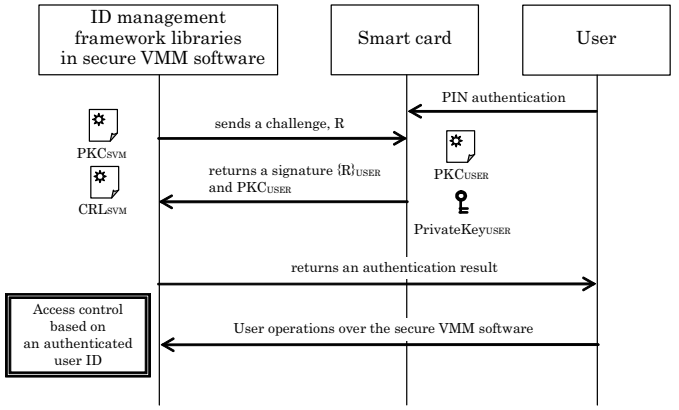


Fig. 4. User authentication between the ID management framework libraries and a user

## 5. Implementation of the Portable ID Management Framework Libraries

We have implemented the portable ID management framework libraries shown in Figure 3. Table 1 shows the hardware and software environment for the implementation. Because the portable ID management framework libraries have middleware software and device drivers completely, the libraries can run on the hardware directly without the help of operating systems and specific external libraries. As a result, we can integrate our portable ID management framework libraries into any VMM and hypervisor software. We employ eLWISE smart card manufactured by NTT Communications. The eLWISE smart cards are employed as Japanese national ID cards (i.e. the Basic Resident Register system in Japan). We have implemented secure VMM libraries compatible with PKCS#11 interfaces. We have merged our portable ID management framework libraries with BitVisor.

## 6. BitVisor

In section 1, we have described BitVisor briefly. BitVisor is a bare metal hypervisor. In the following sections, we use not *VMM* but *hypervisor* because BitVisor does not need host operating system. For the same reason, we also use not *secure VMM* but *secure hypervisor* in the following sections. BitVisor employs CPUs and chipsets supporting Intel Virtualization Technology (Intel VT) or AMD-V. We have presented a detailed design and implementation of BitVisor in the previous paper (Shinagawa et al., 2009). BitVisor supports both 32 bit and 64 bit mode. Current version of BitVisor can run Windows XP/Vista, Linux, and FreeBSD as a guest OS. BitVisor can run only one guest OS at the same time. However, BitVisor achieves very small code sizes and light weight performance. These characteristics are suitable for a

foundation of security policy enforcement. This section shows highlights of BitVisor architecture.

| | |
|---|---|
| Smart card | NTT Communications eLWISE (ISO/IEC 14443 and 7816) |
| Smart card reader | Axalto Reflex USB v3 (ISO/IEC 7816) |
| PC | Intel VT or AMD-V support |
| Hypervisor | BitVisor |
| Compiler | gcc |

Table 1. Software and hardware used in the implementation

First, why do we need to develop a novel hypervisor for security purpose? The purpose of BitVisor is to provide a foundation of security policy enforcement for end-point client computers. There are several implementations of general-purpose hypervisors like Xen and VMware. Although these hypervisors provide general purpose and stable VM execution environments, they are not specialized for security purpose. BitVisor is originally designed for security purpose. If a hypervisor has vulnerabilities, attackers compromise the hypervisor and gain the control of the secure hypervisor software. Therefore, we have to prevent attacks to hypervisors. In general, the large code size of hypervisor causes poor testability and much vulnerability. For instance, VMKernel of VMware ESX server has 200 KLOC (Kilo-lines of codes) (VMware, 2005) and Xen hypervisor has 100 KLOC (Murray et al., 2008). BitVisor (version 0.8) has only 30 KLOC. BitVisor employs a two-step execution mechanism to reduce the code size of core hypervisor (Hirano et al., 2009). Thin hypervisors like BitVisor provide the low risk of vulnerabilities and high testability.

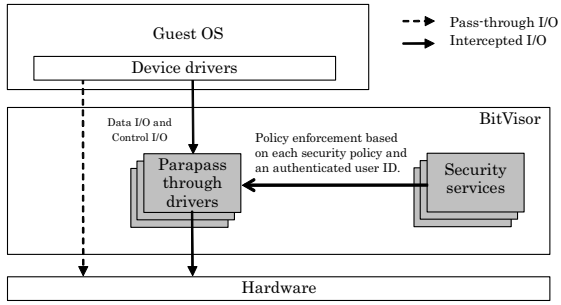


Fig. 5. Parapass-through architecture of BitVisor

Figure 5 shows basic architecture of BitVisor. BitVisor employs parapass-through architecture to process I/O devices. In this architecture, most accesses from a guest OS are passed through BitVisor simply. Therefore, it can provide high performance like native hardware. Some I/O data from specific devices are intercepted by parapass-through drivers. BitVisor can hook I/O data and apply security policies in conjunction with the parapass-through drivers. BitVisor can enforce security services like a storage encryption function and a VPN function based on each security policy. Intercepted I/O are divided into data I/O

and control I/O. BitVisor can intercept control I/O and record it to audit logs in the hypervisor layer. BitVisor can intercept data I/O to inspect and encrypt them.

A parapass-through driver of BitVisor has essential codes only to enforce security services. A device driver on the guest OS executes most tasks other than security processing. Therefore, a parapass-through driver can keep the small code size. To employ a new device to monitor and encrypt, BitVisor needs this small parapass-through driver and its security service library only. Other general purpose hypervisors need much larger code sizes of the new device because of its architecture.

BitVisor also intercepts memory accesses using shadow paging to isolate memory area o f each device. BitVisor employs a novel scheme called shadow DMA (Direct Memory Access) descriptors. Shadow DMA descriptors enable us to intercept data transferred by a DMA mechanism. By employing shadow DMA descriptors, we can prevent attacks using a DMA mechanism from guest OSs.


Fig. 6. Start screen of BitVisor

Figure 6 shows a start screen of BitVisor. BitVisor has been developed by 6 universities and college, and many companies with the help of NISC (National Information Security Center), Japan. BitVisor can be downloaded from http://sourceforge.net/projects/bitvisor/.

## 7. Integration of Portable ID Management Framework and BitVisor

We have integrated the proposed portable ID management framework libraries to BitVisor. The portable ID management framework libraries are not dependent on other external libraries and system calls of host operating systems. Therefore, we can integrate the ID management framework libraries into BitVisor directly. The libraries can run with bare metal hypervisors on the hardware directly without the help of host operating systems.

Figure 7 shows the relation between the proposed portable ID management framework libraries and security services of BitVisor. The proposed ID management framework provides the following functions to BitVisor: (1) a PKI-based user authentication function

using smart card to start a BitVisor system (a simple ID/password authentication function is also provided), (2) a storage function of encryption keys in smart cards (BitVisor provides an XTS-AES storage encryption service), (3) a PKI-based user authentication function using smart cards for IPsec-VPN/IKEv1 service, and (4) a periodical checking mechanism to verify user's presence. The final feature can be used to detect illegal physical access to the end-point computers. If a user removes her or his smart card from the smart card reader, BitVisor detects it and immediately shuts down the BitVisor system.

The portable ID management framework libraries support both contact and contact-less smart cards (ISO/IEC 7816 and 14443) for BitVisor. As shown in Figure 3, the portable ID management framework libraries are constructed as layered libraries. If other service provider needs to handle other smart card and reader products, the provider has to develop a new CCID driver. If a provider needs to customize partitions of smart cards, the provider also has to develop customized PKCS#11 libraries.

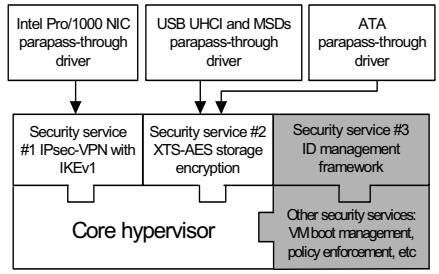


Fig. 7. Portable ID management framework libraries and security services of BitVisor

## 8. Two-step Execution Mechanism for Thin Secure Hypervisors

This section shows a method to minimize the core portion of BitVisor software. A trusted computing base (TCB) is a component to enhance the security of existing operating systems. We employ hypervisors to construct a TCB. In general, the complexity of hypervisors is not preferable to construct a TCB. Some researchers have proposed tiny hypervisors specialized for security-purpose. Murray et al. propose a mechanism to reduce the complexity of Xen hypervisor to construct a TCB (Murray et al., 2008). SecVisor is developed as a tiny hypervisor that ensures code integrity for commodity OS kernels (Seshadri et al., 2007). SecVisor has small code size, only 1,112 LOC (Lines of code), for the run-time portion using CPU-supported virtualization. Xia et al. shows a small hypervisor called Palacious VMM (Xia et al., 2008). Palacious VMM hooks I/O operations between device drivers on a guest OS and physical hardware. The core of Palacious VMM has 20 KLOC and the additional part to hook I/O operations has 10 KLOC. As described above, the code size of a TCB is one of the important aspects to evaluate whether reliable security mechanisms or not.

We have proposed the two-step execution mechanism in the previous paper (Hirano et al., 2009). Figure 8 shows the flow of the proposed two-step execution mechanism. Our proposal is intended to reduce the code sizes of the run-time portion of BitVisor as possible. Basic idea is simple, we separate a conventional hypervisor-based TCB into the following two parts: (1) a thin hypervisor with minimum security services and (2) a special guest OS

for security preprocessing. Thus, we introduce an additional TCB domain for security preprocessing as the special guest OS runs on a hypervisor.

We can move the many tasks from BitVisor to the special guest OS. For example, the special guest OS can execute the following tasks before booting a target guest OS to be protected: (a) a PKI-based user authentication function using a PKI-based smart card for VM boot operations, and (b) an acquisition function of encryption keys for future encryption services from a smart card, and (c) an acquisition function of user certificates for VPN services from a smart card. We have also designed a data passing part between the special guest OS and BitVisor using *kexec* system call of Linux OS. By employing the proposed two-step execution mechanism, we have reduced approximately 8.5 % of LOC in run-time portion of BitVisor in total. Especially, we have reduced 24.5 % of LOC of ID management framework libraries.

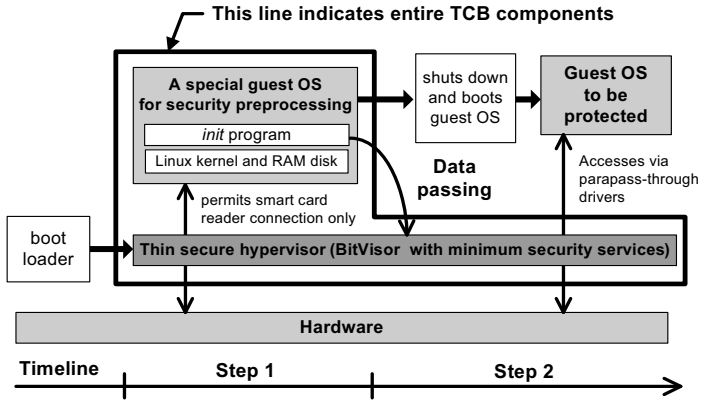


Fig. 8. Flow of the proposed two-step execution mechanism for thin secure hypervisors

To guarantee the authenticity of the special guest OS (the special guest OS consists of the minimum Linux kernel, initial RAM disk and init program only), we can introduce a measured launch mechanism based on a TCG extended boot loader (Sailer et al., 2004) like TrustedGrub. We can also guarantee the authenticity of BIOS codes and the first part of TCG extended boot loader using Intel TXT (Trusted Execution Technology) hardware and TPM chip.

## 9. Future Direction

We have described a basic concept of BitVisor and its portable ID management framework. From the perspective of IT resource management, these basic TCB components can be used as a policy enforcement mechanism for distributed end-point computers in organizations. Current version of BitVisor provides an XTS-AES storage encryption function and an IPsec-VPN function to prevent information leak cases. We must consider further security services to prevent information leak cases via other I/O devices. Policy management problem and certificate management problem are also important to deploy the proposal to real governmental and commercial organizations. Moreover, we need further on-site verifications of the prototype implementation. We have to improve the source codes continuously to prevent attacks to their potential vulnerabilities and increase the usability.

## 10. Conclusion

This chapter has shown the portable ID management framework for secure hypervisors. We have introduced an architectural overview of the novel secure hypervisor software called BitVisor. We have shown the design and the prototype implementation of the portable ID management framework libraries and BitVisor. The source codes of BitVisor including the portable ID management framework libraries can be downloaded from the following web sites, http://sourceforge.net/projects/bitvisor/.

## 11. References

Garfinkel, T.; Pfaff, B.; Chow, J.; Rosenblum, M. & Boneh, D. (2003). Terra: a virtual machine-based platform for trusted computing, *Proceedings of the ACM Symposium on Operating Systems Principles*, pp. 193–206

Hirano, M.; Okuda, T.; Kawai, E. & Yamaguchi, S. (2007). Design and Implementation of a Portable ID Management Framework for a Secure Virtual Machine Monitor, *Journal of Information Assurance and Security,* Vol.2, No.3, pp.211-216

Hirano, M.; Shinagawa, T.; Eiraku, H.; Hasegawa, S.; Omote, K.; Tanimoto, K.; Horie, T.; Mune, S.; Kato, K.; Okuda, T.; Kawai, E. and Yamaguchi, S. (2009). A Two-step Execution Mechanism for Thin Secure Hypervisors, I*n Proceedings of the 3rd International Conference on Emerging Security Information, Systems and Technologies*

Madnick, S. & Donovan, J. (1973). Application and analysis of the virtual machine approach to information system security and isolation, *Proceedings of the workshop on virtual computer systems*, pp.210-224, ACM Press

Meushaw, R. & Simard, D. (2000). NetTop: Commercial Technology in High Assurance Applications, *National Security Agency Tech Trend Notes*, pp. 3-9

Murray, D. G.; Milos, G. and Hand, S. (2008). Improving Xen security through disaggregation. *In Proceedings of the Fourth ACM SIGPLAN/SIGOPS international Conference on Virtual Execution Environments*, pp.151-160

Sailer, R.; Zhang, X.; Jaeger, T. and Doorn, L. V. (2004). Design and Implementation of a TCG-based Integrity Measurement Architecture, *In Proceedings of thirteenth USENIX Security Symposium*, pp.223-238

Sailer, R.; Jaeger, T.; Valdez, E.; Caceres, R.; Perez, R.; Berger, S.; Griffin, J. & Doorn, L. (2005). Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor, Proceedings of *ACSAC 2005*, IEEE CS, pp. 276-285

Seawright, L. & MacKinnon, R. (1979). VM/370 - a study of multiplicity and usefulness, *IBM Systems journal*, pp.4-17

Seshadri, A.; Luk, M.; Qu, N. and Perrig, A. (2007). SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. *In Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles*, pp.335-350

Shinagawa, T.; Eiraku, H.; Tanimoto, K.; Omote, K.; Hasegawa, S.; Horie, T.; Hirano, M.; Kourai, K.; Oyama, Y.; Kawai, E.; Kono, K.; Chiba, S.; Shinjo, Y. & Kato, K. (2009). BitVisor: A Thin Hypervisor for Enforcing I/O Device Security, *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution sEnvironments*, pp.121-130

Xia, L.; Lange J. and Dinda, P. A. (2008). Towards Virtual Passthrough I/O on Commodity Devices., USENIX Workshop on I/O Virtualization


**Engineering the Computer Science and IT**

Edited by Safeeullah Soomro

ISBN 978-953-307-012-4

Hard cover, 506 pages

**Publisher** InTech

**Published online** 01, October, 2009

**Published in print edition** October, 2009

It has been many decades, since Computer Science has been able to achieve tremendous recognition and has been applied in various fields, mainly computer programming and software engineering. Many efforts have been taken to improve knowledge of researchers, educationists and others in the field of computer science and engineering. This book provides a further insight in this direction. It provides innovative ideas in the field of computer science and engineering with a view to face new challenges of the current and future centuries. This book comprises of 25 chapters focusing on the basic and applied research in the field of computer science and information technology. It increases knowledge in the topics such as web programming, logic programming, software debugging, real-time systems, statistical modeling, networking, program analysis, mathematical models and natural language processing.



**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821