Operating System Mechanisms for TPM-Based Lifetime Measurement of Process Integrity

Xiao Li, Wenchang Shi, Zhaohui Liang, Bin Liang, Zhiyong Shan
Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University), Ministry of Education, Beijing 100872, China
School of Information, Renmin University of China, Beijing 100872, China
wenchangshi@ieee.org

Abstract

Implementing runtime integrity measurement in an acceptable way is a big challenge. We tackle this challenge by developing a framework called Patos. This paper discusses the design and implementation concepts of our operating system mechanisms for runtime process integrity measurement, which is an important part of the Patos framework and is named Patos-RIP. Patos-RIP is developed into the main-stream Linux operating system and utilizes TPM as hardware support for tamper-resistance. From the beginning a process is created to the moment the process dies, Patos-RIP conducts integrity measurement at appropriate points of time when the process runs, so as to ensure that the integrity of a process is not compromised during its whole lifetime. This way, Patos-RIP can improve trustworthiness of processes by effectively detecting runtime tampering attacks on processes' integrity.

1. Introduction

In a computerized world, trustworthiness of a system depends heavily on integrity of system components, which is potentially threaten by myriad malicious sources, especially in an open, distributed, networked environment.

Recently, pushed by motivations of building trusted computing ecosystems, considerable work has been undertaken in providing system integrity measurement. However, much of this work still shows obvious limitations. Implementing runtime integrity measurement in an acceptable way is still a big challenge. We are endeavoring to tackle this challenge by developing a framework called Patos (Platform Architecture based on Trusted Operating System).

Introducing new functionality in the cost of demanding revolutionary change to legacy application systems is impractical. The philosophy underlining Patos is to implement light-weight trust establishment mechanisms with effective compatibility for existing application.

This paper discusses the design and implementation concepts of our operating system mechanisms for runtime process integrity measurement, which is an important part of the Patos framework and is named Patos-RIP (Runtime Integrity of Process). Patos-RIP is developed into the main-stream Linux operating system and utilizes TPM (Trusted Platform Module) as hardware support for tamper-resistance. From the beginning a process is created to the moment the process dies, Patos-RIP conducts integrity measurement at appropriate points of time when the process runs, so as to ensure that the integrity of a process is not compromised during its whole lifetime.

2. System Architecture Overview

Patos-RIP is devised to be an integral part of an operating system. It augments to the operating system functionality of measuring process integrity at runtime. Briefly speaking, an architectural overview of Patos-RIP can be depicted as Figure 1.

The Patos-RIP architecture may be divided into three parts, namely Division-F, Division-N and Division-P, which will be explained in the following sub-section.

2.1. Key Components

Patos-RIP makes use of three repositories to hold information necessary for process integrity measurement. They are the OffBank (Off-Line Integrity Characteristics Bank), the OnBank (On-Line Integrity Characteristics Bank), and the Mass Storage. The OffBank stays in a trusted storage. It holds integrity characteristics of all executable images needs

978-1-4244-5113-5/09/$25.00 ©2009 IEEE

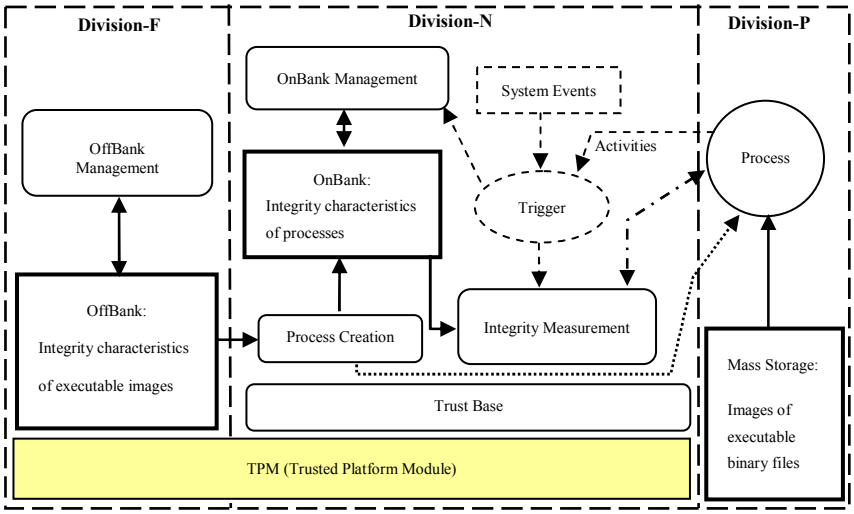

Figure 1. An Architectural Overview of Patos-RIP

to be monitored. The OnBank sits in main memory. It holds integrity characteristics of all processes needs to be measured in a system. The Mass Storage is usually a disk and is managed in the way of a file system. It holds images of all executable binary files from which processes are derived.

Division-F consists of the OffBank and the OffBank Management component. Processes to be measured live in Division-P. The Mass Storage used for processes creation is also located in Division-P.

Division-N is the core of the Patos-RIP. It contains the Integrity Measurement component, the Process Creation component, the OnBank Management component, as well as the Trust Base component. The Trust Base is the fundamental part of an operating system kernel that provides basic integrity measurement support. Of course, the OnBank also falls into this division.

Both Division-F and Division-N are dependent on the TPM hardware to ensure trusted functionality.

2.2. Working Principles

The main task of the Patos-RIP is to check a process' integrity whenever it is necessary after the process is born and before it dies. Systems events and processes' activities may trigger the Integrity Measurement component to work. When the Integrity Measurement component is working, it searches integrity characteristics for the process from the OnBank. If there is a hit, current integrity characteristics will be generated from the process on the fly. Then, the newly generated characteristics will be compared with those found from the OnBank. If they match, it is concluded that the process is in good integrity. Otherwise, it says that the integrity of the process has been compromised.

During the lifetime of a process, it is possible for the process to change its integrity characteristics legally. In this case, the OnBank must be updated to reflect the real integrity of a process correctly. It is the OnBank Management component that is responsible for the update of the OnBank. System events or a process' activities that are related to a change to integrity characteristics may trigger the OnBank Management component to fulfill this mission of update.

Initial integrity characteristics in the OnBank for a process are derived from information in the OffBank. Information in the OffBank is of static integrity characteristics, while information in the OnBank is of dynamic integrity characteristics. Given an image of an executable binary file, the OffBank Management component can generate integrity characteristics for it. If the image is trustworthy, integrity characteristics generated from it can be put into the OffBank.

In summary, the Patos-RIP carries out runtime process integrity measurement in the following way:

- As a pre-processing step, the OffBank Management component generates integrity characteristics for a given trusted executable image and save them into the OffBank.
- On creating a process, the Process Creation component sets up the process according the corresponding image in the Mass Storage and establishes its integrity characteristics in the OnBank based on information in the OffBank.

- Whenever it is necessary, the Integrity Measurement component measures the integrity of a process according to information obtained from the OnBank and generated from the process.
- Whenever it is necessary, the OnBank Management component update integrity characteristics in the OnBank for a given process.
- Hardware TPM functionality is used to help protecting the integrity of the OnBank and the OffBank.

2.3. The Threat Model

The main purpose of developing Patos-RIP is to fight against runtime attacks on process integrity. While Patos tries to deal with integrity problems of a whole platform, Patos-RIP focuses on integrity of protected processes. It has its eyes on the behavior of any given process, so as to ensure that the process can keep good integrity during its whole lifecycle.

So long as a process is alive, Patos-RIP keeps checking its integrity at certain critical points of time. Each time Patos-RIP performs checking, only when integrity criteria is well satisfied will it confirm that the process in question is in good integrity condition. No matter whether there are discrepancies between the integrity characteristics in the OnBank and those from the current process, or there is no corresponding entry in the OnBank for a given process, Patos-RIP will claim it an abnormal condition.

Therefore, the power of Patos-RIP is to determine if a pre-registered process is tampered with at runtime, and to determine if a running process is a pre-defined legal one.

3. Integrity Measurement Mechanisms

Integrity measurement mechanisms of Patos-RIP exist mainly in Division-N. They make use of the OffBank in Division-F and the Mass Storage in Division-P. The measurement targets of them are processes in Division-P.

3.1. Integrity Characteristics of a Process

The body of a process is a program, which is expressed as an executable binary image. A program consists of text and data. Each process has a context. Not only the program but also the context has impact on the integrity of a process.

Patos-RIP regards the integrity of a process as the integrity of its program together with that of its context. It defines integrity metric in term of cryptographic hash, or fingerprint. As a consequence, integrity characteristics of a process comprise fingerprints of its program and its context, while integrity characteristics of a program include fingerprints of its text and its data.

In a Linux system, the most widely used format of executable files is ELF (Executable and Linking Format). For brevity of description and without loss of generality at the same time, this paper only discusses ELF files, and it assumes that the Mass Storage is a disk managed with the ext2 filesystem.

By analyzing an ELF file, the OffBank Management component can figure out the text and the data of a program. Then, it collects fingerprints of the text and the data for the program, and stores them in the OffBank. For each process whose integrity is to be monitored, a trusted copy of its program image should be provided to the OffBank Management component to collect fingerprints. Each protected image file is assigned in the OffBank an entry that holds text fingerprints and data fingerprints of the image. Each entry in the OffBank is identified by a unique Image Fingerprint Identity (IFID) of the corresponding image file.

In addiction to image fingerprints, context fingerprints form an important part of a process' fingerprints. All together, in the OnBank, image fingerprints and context fingerprints of a process are held in an entry that is identified by a unique Process Fingerprint Identity (PFID) of the process. When a process is created, the Process Creation component copies the image fingerprint from the OffBank entry into the OnBank entry, and collects and fills the context fingerprint into the same OnBank entry.

The OnBank entry is used as the baseline for measuring the integrity of a process. The OnBank Management component is in charge of updating the OnBank entry of a process to reflect its real state.

3.2. Measuring the integrity of a process

Patos-RIP's mission of measuring the integrity of a process is fulfilled by the Integrity Measurement component. The moments in the lifecycle of a process for the Integrity Measurement component to measure the integrity of the process may be:

- When the process is born;
- When the process is to run its code (text);
- When the process is to access its data;
- When the process is switched back to CPU either due to scheduling or interrupt.

The basic action taken to measure integrity is to compare the pertinent fingerprint retrieved from the OnBank entries with that collected at the moment.

Integrity measurement of text or data is carried out in the unit of page. One page a time. The fingerprint of

text or data is collected from the main memory. When a page of text or data is loaded into memory, its integrity is checked. Only when the page of content passes the check successfully may the process go on working normally. Otherwise, Patos-RIP issues a warning that the process is not in a good integrity state. Further action may be taken as necessary.

Context of a process is mainly defined by a set of registers holding information critical to the process. Among others, ESP, EIP and CR3 are examples of important registers. Collecting context fingerprint means combining the contents of these registers and computing cryptographic hash of the combined result. It is not necessary to keep a fingerprint for every register. One fingerprint of a collective content for all registers is enough.

In response to scheduling or interrupt, the running of a process is suspended, and the process' context is saved. At a later time, when the process is to get running again, its context has to be restored. At this point of time, Patos-RIP checks the fingerprint of the context, in order to ensure that its integrity is not impaired in the interval of suspension of the process.

3.3. Fundamental of System Design

The OffBank and the OnBank are two important repositories used as baseline for process integrity measurement in Patos-RIP. Among other things, establishing and maintaining the OffBank and the OnBank are crucial work for Patos-RIP to do.

The OffBank is made up of a set of OffBank entries. The layout of an OffBank entry is described in Figure 2.

For each executable file, one OffBank entry is created to record its integrity characteristics, namely, fingerprints. Given a specific executable file, whose integrity needs to be measured at runtime, the OffBank Management component figures out its Text part and Data part. Both the Text part and the Data part are divided into fixed-length pages. The OffBank Management component computes the hash value of each page and save it into the OffBank entry. The i-th page of the Text part is signified as cPage-i, and the corresponding hash value as hCode-i. Similarly, dPage-j for the j-th page of Data, and hData-j its hash value.

Suppose that the Text part is m pages long and the Data part is n pages long, in the OffBank entry, there will be m fingerprints of Text and n fingerprints of Data correspondingly. The number of Text fingerprints is recorded with the hCode_count field of the OffBank entry, and that of Data fingerprints the hData_count field.

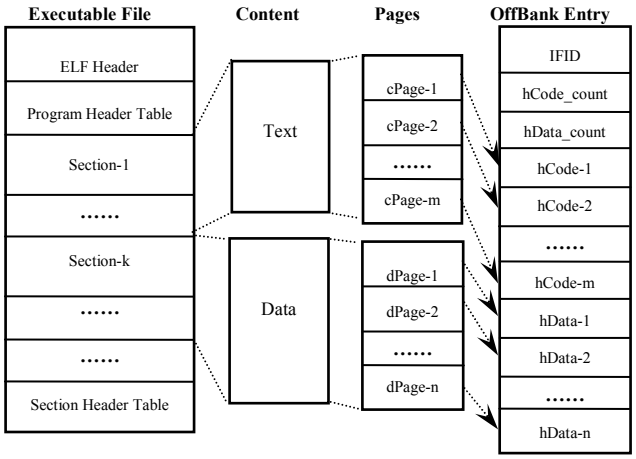

Figure 2. Layout of the Off-Line Bank Entry

Each executable file is given an IFID (Image Fingerprint Identity) to identify its uniqueness, which is save in the IFID field of the OffBank entry.

The OnBank is devised as a table consisting of a serial of OnBank entries. The layout of an OnBank entry is expressed with Figure 3.

From the perspective of what it holds, an OnBank entry is very much like an OffBank entry. The principal elements of an OnBank entry are fingerprints of Code and Data of a process, each of which corresponds to a page of Code or Data. Additionally, an OnBank entry holds the fingerprint of the context of a process with a field named hContext.

Initially, fingerprints of Code and Data in an OnBank entry are copied from a correlated OffBank

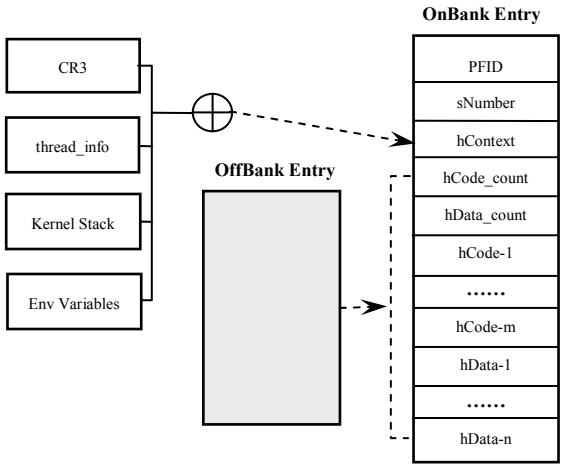

Figure 3. Layout of the On-Line Bank Entry

entry when a process starts its life. When a process modifies some contents in a page legitimately, the pertinent fingerprint of the page will be updated. Obviously, while an OffBank entry is static, an OnBank entry is dynamic in the sense of experiencing change during the lifetime of a process.

Since the OnBank is organized as a table, as an entry of a table, an OnBank entry is of fixed-length. The number of fields in an OnBank entry used to hold fingerprints of Code and Data is defined as NBE_MaxHash. If the result of adding hCode_count to hData_count is greater than NBE_MaxHash, more than one OnBank entries are needed for a process. A field named sNumber is used to support this situation. For a process that only needs one OnBank entry, sNumber is set to zero. For a process that consumes more than one OnBank entries, sNumber takes a positive integer value to represent the sequent number of an entry.

The context of a process is mainly defined by values of registers. In collecting the fingerprint of the context, environment variables of a process are also considered.

The CR3 register holds the physical address of the PGD (Page Global Directory), which is very important for virtual address to physical address translation. In response to context switching, some special purpose registers such as the stack pointer register, ESP, are save to the thread_info area, and various general purpose registers such as EAX, EBX, ECX, and EDX, etc., are save to the kernel stack area. Figure 3 categorizes registers with this respect.

Each process in question is given a PFID (Process Fingerprint Identity). The combination of PFID and sNumber uniquely identifies an OnBank entry, which in turn provides the fingerprint in need.

3.4. Considerations of System Implementation

Patos-RIP is implemented in a Linux system. It is natural to think of determining measurement points in the Linux system and utilizing them to invoke measurement operations.

A process starts its real life with system call execve(), in response to which, a serial of kernel function calls are invoked to get the image file:

sys_execve() → do_execve() → open_exec() → nameidata_to_filep() → dentry_open() → open(inode)

Then, the following kernel function calls are invoked to load the program into the process' virtual address space:

search_binary_handler() → load_elf_binary() → elf_map() → do_mmap() → do_mmap_pgoff() → get_unmapped_area()

Basic memory mapping of Code and Data is established for the process. But no Code or Data is loaded into physical memory yet. This is a good point to create OnBank entries for the process.

When the CPU executes an instruction of the process, a page fault interrupt will occur. The following kernel function calls are invoked to bring the page of code in need into physical memory:

do_page_fault() → handle_mm_fault() → do_no_page()

It is time to measure the integrity of the page of code. With the page of code in physical memory, it is easy to collect its instant fingerprint by hash calculation. On the other hand, according to the virtual address of the instruction, the virtual page number of code, say i, can be figured out. Therefore, the saved fingerprint, or hCode-i, can be retrieved from the related OnBank entry. If the instant fingerprint is different from hCode-i, a warning is issued that the integrity of the process is impaired.

Reading or writing data of a process may cause page fault interrupt too. At the time kernel function do_no_page() works, data integrity measurement may be conducted by collecting instant fingerprint of data page and retrieving the corresponding saved fingerprint hData-j. In conjunction with data integrity measurement, data fingerprint in an OnBank entry is updated after a legitimate write operation.

Context integrity measurement and context fingerprint update are in need when context switching occurs.

Process scheduling performed by kernel function schedule() causes context switching. When a context is switched out, its fingerprint needs to be updated. If a switched in context is one that was previously switched out, its integrity is measured.

Interrupts causes context switching as well. When an interrupt occurs, kernel function do_IRQ() is invoked. It saves all registers into the stack SS0:ESP0, suspends the current process, and executes the interrupt handler. After the interrupt handler finishes its work, registers are restored from the stack SS0:ESP0, and the suspended process gets running again.

When a process ends its life with system call exit(), information in its OnBank entries is erased, and the OnBank entries are released.

3.5. Protection with TPM

It is very difficult, if not impossible, to implement full tamper-resistance with pure software methods. Patos-RIP depends on the widely available TPM hardware to provide protection against tampering attacks. A TPM is a tiny hardware device that is powerful for integrity measurement, storage and report.

The trustworthiness of the OffBank and the OnBank is essential for making decision about the integrity of a

process, because they are used as baseline for integrity measurement.

The OffBank may be put on a separate computer with TPM support, which can be called an OffBank platform. The OffBank platform is configured as a relatively closed system that is not used for network service except for communicating with the other part of Patos-RIP. The OffBank is encrypted in association with a well-defined platform configuration through the Sealed Storage function of the TPM. Only on the exact configuration can the OffBank be decrypted. Hence its integrity can be ensured.

With trustworthiness of the OffBank, the integrity of the initial OnBank can be well established. To keep the integrity of the OnBank later on, OnBank entries are encrypted with TPM encryption functions. The TPM generates a pair of keys for each process, which can be placed along with the OnBank entries of the process. The OnBank entries of a process are encrypted with the public key of the process. Private keys of all processes are encrypted by the TPM with its internal SRK (Storage Root Key). The encryption of a process' private key is bound to the PFID of the process so that a malicious process with incorrect PFID can not successfully ask the TPM to decrypt OnBank entries of other processes. To decrypt its own OnBank entry, a process loads its encrypted private key into a key slot of the TPM. The TPM decrypts the process' private key with consideration to its PFID, and decrypts the encrypted OnBank entry for it. All decryption operations are carried out inside the TPM.

Not only integrity of OffBank and OnBank but also that of all Patos-RIP components needs to be ensured. Patos establishes essential trustworthiness for the basic Linux kernel with a TPM-based trusted boot procedure. Within the basic Linux kernel is the Trust Base of Patos. The Trust Base supports integrity measurement of Patos-RIP components.

Collecting fingerprints of would-be-measured targets is one of the core operations that Patos-RIP performs. It uses hash functions of the TPM to accomplish this job wherever possible.

4. Related Work

Providing system integrity measurement to establish trustworthy application environment has been being attracting considerable work for a long time.

Dyad[1], BITS[2], UPen AEGIS[3], IBM 4758[4], TPod[5] and IMA[6] tried to build up trustworthy environments by way of trusted boot. Trusted boot ensures that a successfully booted system is always a trusted system. UPen AEGIS and IBM 4758 also implement secure boot. Tripwire[7] provides an early framework for integrity measurement. It provides assurance that system files are up-to-date and have not been tampered with. XOM[8] tries to build a trusted platform using traditional time-sharing operating system executing on a processor architecture that provides copy protection and tamper-resistance functions. MIT AEGIS[9] implements integrity measurement mechanisms inside a secure processor and provides process integrity measurement support. In addition to support trusted boot, IMA uses the TPM to detect subversion of the measurement results by comparing a hash value stored in the TPM with the expected value generated from the measurement system's audit log. PRIMA[10] extends the IMA concept to better minimize the performance impact on the system. By coupling IMA to SELinux policy, the number of measurement targets can be reduced to those that have information flow to trusted objects. Copilot[11] uses cryptographic hashes to detect changes in measured objects, but its target of measurement is not the static image of a program and configuration files but the memory image of a running system. NFORCE[12] measures the in-core image of the Linux operating system and its executing processes. It uses the SMM feature of the Intel CPU architecture to provide a protected execution environment from which to measure. Its measurements are still limited to static memory regions in predefined locations. LKIM[13] employs contextual inspection as a means to more completely characterize the operational integrity of a running Linux kernel. It examines dynamic data structures to provide improved integrity measurement. Patagonix[14] is a hypervisor-based system that depends on the processor hardware to detect code execution and on the binary format specifications of executables to identify code and verify code modifications.

Our work is different from those of others in that it tries to build light-weight operating system mechanisms for runtime process integrity measurement based on widely available TPM hardware. MIT AEGIS is most closely related to our work in providing lifetime process integrity support, but it depends heavily on proprietary secure processor. Patagonix is of certain similarity to our work in integrity measurement framework, but it chiefly focuses on hypervisor support and does not consider context integrity measurement.

5. Conclusions

This paper presents the design and implementation concepts of Patos-RIP, a prototype system of operating

system mechanisms for runtime process integrity measurement. Contributions of our work include:

a) Demonstrate an approach to implementing light-weight mechanisms for runtime process integrity measurement that can provide effective compatibility for existing applications;

b) Push one step forward in supporting process lifetime integrity measurement without reliance on strong proprietary hardware support that may sacrifice system flexibility;

c) Push one step forward in utilizing widely available hardware functions to enhance trust and security functionality of main-stream software system.

Both effectiveness test and performance penalty test are crucial to evaluate the practicability of a system. Due to space limitation, this part of work is not presented in this paper. We will discuss such kind of test of Patos-RIP in detail in another paper. We will not only test Patos-RIP alone but test Patos as a whole.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. The work of this paper is supported by National 863 High-Tech Research Development Program of China (2007AA01Z414), National Natural Science Foundation of China (60873213, 60703103) and Natural Science Foundation of Beijing (4082018).

References

[1] J.D. Tygar, B. Yee. Dyad: A System for Using Physically Secure Coprocessors. Technical Report, CMU-CS-91-140R, Carnegie Mellon University, May 1991.

[2] P.C. Clark, L.J. Hoffman. BITS: A Smartcard Protected Operating System. Communications of the ACM, 37(11), Nov. 1994: 66~70, 94.

[3] W.A. Arbaugh, D.J. Farber, J.M. Smith. A Secure and Reliable Bootstrap Architecture. Proceedings of the 1997 IEEE Symposium on Security and Privacy (S&P'97), 1997: 65~71.

[4] J.G. Dyer, M. Lindemann, R. Perez, R. Sailer, L.V. Doorn, S.W. Smith, S. Weingart. Building the IBM 4758 Secure Coprocessor. IEEE Computer, 34(10), Oct. 2001: 57~66.

[5] H. Maruyama, F. Seliger, N. Nagaratnam, T. Ebringer, S. Munetoh, S. Yoshihama, T. Nakamura. Trusted Platform on demand. Technical Report, RT0564, IBM, Feb. 2004.

[6] R. Sailer, X. Zhang, T. Jaeger, L. Van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA, Aug. 2004: 223~238.

[7] G.H. Kim, E.H. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. Proceedings of the 2nd ACM Conference on Computer and Communication Security, Fairfax, Virginia, USA, 1994:18-29.

[8] D. Lie, C.A. Thekkath, M. Horowitz. Implementing an Untrusted Operating System on Trusted Hardware. ACM SIGOPS Operating Systems Review, 37(5), Dec 2003:178~192.

[9] G.E. Suh, D. Clarke, B. Gassend, M. van Dijk, S. Devadas. AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing. Proceedings of the 17th Annual International Conference on Supercomputing (ICS'03), ACM Press, 2003:160~171.

[10] T. Jaeger, R. Sailer, U. Shankar. PRIMA: Policy-Reduced Integrity Measurement Architecture. Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies, Lake Tahoe, California, USA, ACM Press: New York, NY, USA, 2006: 19~28.

[11] N. Petroni Jr., T. Fraser, et al. Copilot - a coprocessor-based kernel runtime integrity monitor. Proceedings of the 13th Usenix Security Symposium, Aug., 2004:179~194.

[12] D. Heine and Y. Kouskoulas. N-force daemon prototype technical description. Technical Report VS-03-021, The Johns Hopkins University Applied Physics Laboratory, Jul., 2003.

[13] P.A. Loscocco, P.W. Wilson, J.A. Pendergrass, C.D. McDonell. Linux Kernel Integrity Measurement Using Contextual Inspection. Proceedings of the 2007 ACM workshop on Scalable trusted computing (STC'07), 2007:21-29.

[14] L. Litty, H.A. Lagar-Cavilla, D. Lie. Hypervisor Support for Identifying Covertly Executing Binaries. 17th USENIX Security Symposium, 2008:243–258.