Research Article

Research on Dynamic Integrity Measurement Model Based on Memory Paging Mechanism

Chaowen Chang, Xin Chen, Shuai Wang, and Qinghai Xiao

Zhengzhou Institute of Information Science and Technology No. 307, Henan 450004, China

Correspondence should be addressed to Chaowen Chang; changchaowen5@163.com

Received 2 November 2013; Accepted 10 December 2013; Published 9 January 2014

Academic Editor: Guoliang Wei

Copyright © 2014 Chaowen Chang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In order to solve the existed problems of dynamic integrity measurement method, a dynamic integrity measurement model based on Memory Paging Mechanism is proposed in this paper. The model takes memory pages of executable subjects as measurement objects. When the pages are scheduled into memory, the measurement points are inserted, the pages are measured, and their integrities are verified. The model is able to insure the integrity and trust of each executable page and assure that the integrity of the whole executable subjects is not destroyed. To verify this model, XEN hypercall mechanism is used to acquire executable subjects' pages scheduled into memory, and the integrity measurement and verification codes are put into hypercall handler. Accordingly, dynamic integrity measurement to executable subjects is implemented.

1. Introduction

Trusted computing technology is derived from safety guidance file of Rainbow Series information system launched by the USA in 1983. In 1999, TCG firstly presented the concepts of Trusted Computing Platform (TCP) and Trusted Platform Module (TPM) and proposed the specific structure and technology route of TCP system [1].

The cores of trusted computing technology are trusted computing base and trusted chain [2, 3], and trusted measurement is a key problem of this technology [4, 5]. Trusted computing puts the integrity as a fundamental attribute of trust. The reliability of the platform is depending on whether the integrity measurement value is equal to the corresponding integrity reference value.

The integrity measurement can comprise static measurement and dynamic measurement. The objects of static measurement are executable parts loaded into the platform during startup, such as BIOS, OS Loader, and operating system kernel. The objects of dynamic measurement are executable subjects, including code, data, and library file. The static measurement is a fixed sequence of single-chain process [6], and the measurement process is still unchanged when the

operations of the platform change. Different from the static measurement, the process of dynamic measurement is multiple and unordered, and the time and space of object loading and running are not fixed. Obviously, the dynamic measurement is more accurate to describe and evaluate the integrity of the platform, and it can also get more attention and research [7].

Integrity Measurement Architecture (IMA) [8] is a well-known integrity measurement structure developed by IBM. Based on the measurement methods of TCG, IMA expands the content of integrity measurement and uses an executable subject, dynamic loader, kernel module, and dynamic link library as measurement objects to measure the integrity of executable subject. Policy Reduced Integrity Measurement Architecture (PRIMA) [9] proposes a policy-reduced integrity measurement structure to solve the problem of loading time measurement.

The Secure Bus (SB) [10] is a trusted computing architecture proposed by George Mason University. The architecture uses TPM as a trusted root and adds a Secure Kernel (SK) and a SB. The trusted relationship is transmitted in accordance with the order of hardware, SK, operation system, SB, and executable subject. SK is located between the hardware and

operation system kernel and isolates the program processes from the underlying hardware. SB performs a dynamic integrity measurement for each program process that makes a hash operation for input and output data and makes a signature for output data. A new embedded measurement module has been designed based on SB measurement method [11]. This module can make the computing platform with nontrusted components use trusted computing functions and keep the platform operating well.

In general, the evaluation indexes of measurement methods are measurement scope and implementation mechanisms [12]. IMA measurement has great performance and efficiency, and it can perform measurement during process loading. However, IMA cannot grasp the dynamic state of the process due to the LSM core hook function, so that it cannot perform measurement during process running. At the basis of IMA, PRIMA increases mandatory access control policy, which makes it flexible to control measurement to keep a balance between safety and feasibility [13], but PRIMA still performs measurement only during process loading. Furthermore, PRIMA needs to measure mandatory access control policy and trusted subjects, which increases the difficulty and complexity. The SB method can measure integrity of process running stage preventing from tampering between processes and measure integrity of input and output stage using a hash algorithm for program process code. However, due to the strict process isolation mechanism, the processes running in the SB are very limited. Though the SB model is to ensure security, it reduces the usability, which makes it difficult to popularize.

To solve the existed problems and insufficiency of dynamic integrity measurement method, a dynamic integrity measurement model based on Memory Paging Mechanism (MP) is researched. To prove this measurement model, XEN hypercall mechanism is adopted to acquire executable subject's pages as measuring objects, inserts measurement points, measures and verifies them before the pages are scheduled into memory, and puts measurement codes and verification codes into hypercalls handle function, accordingly to implement dynamic integrity measurement to executable subjects. The contribution in this paper is that the dynamic integrity measurement can be effectively implemented via taking the executable main memory page as measure objects by applying the operating system Memory Paging Mechanism.

Notation. In the paper, the components which need dynamical measurement are running services, processes, or executable subjects, denoted by S , Memory Pages denoted by P , and Storage Block denoted by B .

2. Design of Dynamic Integrity Measurement Model Based on Memory Paging

Memory is the space in which all the codes and data are stored temporarily. The changing of code and data can be observed in memory space. Therefore, if it could be effective to access memory content, dynamic integrity measurement can be implemented effectively.

FIGURE 1: Dynamic integrity measurement model based on memory paging.

2.1. Model Design. In the view of memory management, the subject S in its life cycle can be seen as a dynamic combination of a number of memory pages $P_1, P_2, \dots, P_N$; namely,

$$S = \{P_1, P_2, \dots, P_N\}, \quad (1)$$

where N is the number of memory pages occupied by subject S in the life cycle.

Obviously, if all loaded dynamic pages are integrated, the subject S is dynamically trusted.

Axiom 1. If and only if $\forall P_i (\text{Hash}(P_i) = P_{i0})$, $i = 1, 2, \dots, N$, where P_{i0} is integrity expected value of page P_i , then the subject S is dynamically integrated or trusted.

According to Axiom 1, dynamic trusted measurement for the subject can be equivalent to dynamic loaded page measurement. The MP dynamic measurement model is described in Figure 1.

The MP model mainly includes monitoring module and measurement module.

(1) Monitoring Module. Monitoring module is used for page monitoring and execution controls and mainly consists of page monitoring unit and execution control unit.

The page monitoring unit is responsible for monitoring the physical memory space. When the page monitoring unit adjusts pages, it can get memory pages on time and then send the pages to the measurement module.

The execution control unit is responsible for controlling the execution of the memory page. After finishing the integrity measurement, the execution control unit needs to decide whether executing the page or interrupting the program based on the measurement result.

(2) Measurement Module. Measurement module is used for integrity measurement and mainly consists of integrity computing unit, integrity verification unit, and integrity

reference value. Measurement module communicates with the underlying trusted platform module.

The integrity computing unit gets the integrity value by using a specified hash algorithm to measure the feature information of a page sent by page monitoring unit.

Integrity verification unit gets a check result by comparing integrity value with integrity reference value of a page. The page is trusted if the check result is consistent, otherwise it is not trusted. And then the check result should be sent to monitoring module.

2.2. Integrity Reference Value for Dynamic Page. The integrity reference value is an evidence of integrity verification. This model can dynamically calculate page reference value P_{i0} based on the block mechanism of physical storage.

As we know, each subject's address space is composed of code segment, data segment, and stack segment. However, because the code segment contains the machine instructions of executable programs, it becomes a main target of Trojan virus and other computer viruses. In this paper, integrity measurement method of code segment page is only analyzed.

In general, the computer file system (FAT, FAT32, NTFS, etc.) is usually stored and managed based on block or sector mechanism. Namely, when the subject S is stored physically, it can be seen as a sequence combination of storage block B :

$$S = B_1 \parallel B_2 \parallel \cdots \parallel B_M, \quad (2)$$

where M is a number of physical storage blocks that S occupied.

When the subject S loads and runs, some blocks of S will be loaded to Buffer. According to the management mechanism of file system, a block or several continuous blocks of the subject S are loaded every time. Hence, every Buffer can be expressed as

$$\text{Buffer} = B_b \parallel B_{b+1} \parallel \cdots \parallel B_e, \quad e \geq b, \quad (3)$$

where b is the number of beginning block, and e is the number of ending block.

According to the memory management mechanism of the operating system, memory is paged. In fact, memory is a set of Buffer, and it can be discussed in two specific cases. (1) Page is composed of several Buffers and Buffer is quite small. (2) Page is a part of Buffer and Buffer is quite big.

In the first case, every Buffer can be corresponding with several physical storage blocks of the subject S . Obviously, pages can also be corresponding with the physical storage blocks of the subject S through the Buffer.

As we know, in the operating system, the size of every memory page P is the integer times that of the storage block B . Therefore, when the page is a part of Buffer, the beginning of every page must be corresponding with a certain storage block, and the page can also be corresponding with some physical storage blocks of the subject S through the Buffer.

According to the analysis above, we can get

$$P_i = B_{i(b)} \parallel B_{i(b+1)} \parallel \cdots \parallel B_{i(e)}, \quad e \geq b. \quad (4)$$

Based on (4), we can get the integrity reference value of P_i through some simple hash calculations, which is

$$P_{i0} = \text{Hash}(B_{i(b)} \parallel B_{i(b+1)} \parallel \cdots \parallel B_{i(e)}). \quad (5)$$

It means that we can also get P_{i0} by calculating the integrity reference value of the subject S storage blocks.

Similarly, when the subject S loads a Dynamic Link Library (DLL), the integrity reference values of the pages can be got by calculating that of the subject S and DLL.

2.3. Measurement Point. Selecting the measurement point is a key factor of available measurement model.

In general, when an executable subject is executed, the operating system will create a complete program map for it, including code, data, stack, and parts of library file. At the same time, the operating system will allocate memory, load the current block into memory, and create a new page table for mapping physical address space and virtual address space. When page load fails, the operating system will replace some new needed pages from disk to the memory based on page replacement algorithm and update the page table.

Based on the above analysis, in the whole life cycle of the executable subject S , the operating system manages and operates its memory space. But the change of memory space must cause the page table updated or created. Hence, when the operating system creates or updates the page table, a measurement point should be inserted and carry out the presupposed measurement program. Then, measurement module calls TPM hash algorithm to measure the integrity value for the feature information of executable subject in that page. After that, measurement module should compare the integrity value with integrity reference value. The operating system executes that page if the measurement is successful, otherwise it exits.

2.4. Dynamic Measurement Process. The measurement process of the MP measurement model begins when page loads. The first step is to get page content through related mechanism and then use the integrity computing unit for calculating feature information in the page. The next step is to execute and verify the integrity and then go back to the original page.

The dynamic measurement process of MP model is showed in Figure 2.

Assume that the current execution (or update) page is P_i , $\text{Address}(P_i)$ is P_i 's initial address in the Buffer, and the $\text{Memory_snap}(\text{Address}(P_i))$ is P_i 's actual content. Dynamical measurement algorithm is described as follows.

- (1) Paging unit loads page P_i into memory and executes it. Obviously,

$$P_i \Rightarrow \text{Address}(P_i) \Rightarrow \text{Memory_snap}(\text{Address}(P_i)). \quad (6)$$

- (2) Measurement program obtains this page and uses hash algorithm to calculate the feature information of page frame P_i . Then,

$$V_i = \text{Hash}(\text{Memory_snap}(\text{Address}(P_i))). \quad (7)$$

FIGURE 2: The dynamic measurement process of MP model.

- (3) Measurement program gets the corresponding integrity reference value P_{i0} of the page frame P_i from (5). Then,

$$P_{i0} = \text{Hash}(B_{i(b)} \parallel B_{i(b+1)} \parallel \dots \parallel B_{i(e)}). \quad (8)$$

In (5), (b, e) can be obtained through data structure pointed by Buffer's pointer, and Buffer's pointer is corresponding with P_i .

- (4) The integrity evaluation unit (it is TPM in the general case) compares P_{i0} with V_i . If $V_i = P_{i0}$, page P_i is trusted, and it is allowed to load and run. Otherwise, error handler should be executed.

According to Axiom 1, if and only if $\forall P_i (V_i = P_{i0})$, then we can consider that the system is dynamical, complete, and trusted.

2.5. Model Analysis

(1) *Model Feasibility.* Dynamical integrity measurement model MP basically complies with TCG measurement framework and mechanism model, including two units: integrity measurement and integrity verification. By the way, the function of the two units is consistent with TCG's measurement model. The feasibility of the MP model is decided by inserting mechanism for measurement point. That is, the MP measurement model can be implemented as long as the measurement point is inserted successfully. The memory paging mechanism is a basis of MP measurement model, while it is implemented by the operating system. At present, the research of the operating system is very comprehensive especially the open-source operating system. Through modifying the core of the operating system, it is possible to implement the dynamic MP measure model based on the memory paging mechanism.

(2) *Attack Detection Capability.* Tampering attacks for memory page consist of Load Before Attack (LBA) and Load After Attack (LAA). LBA is attacking the page mapping stored in the disk. That is, the page is tampered before it is loaded

into memory. LAA is to use some malicious processes for tampering the memory after the page is loaded.

The MP measurement model can resist the above attacks. For example, if the model is attacked by LBA, the page will be tampered and the integrity value will be changed, but the model's integrity reference value is never changed. Therefore, when the measurement model executes the verification operation, it must be able to detect that the page has been falsified and then execute the error handler.

If the model is attacked by LAA, tampered memory always causes page replacement, page failure, and page move-out. When the page failure occurs, any tampered pages will be moved to disk, and the execution flow will be interrupted when this case is detected by the measurement program. When the page move-out or page replacement occurs, the memory paging mechanism can firstly find a seldom-used page frame and move this page to disk and load new needed page into the page frame. Then the measurement program can find that the memory page is tampered. The malicious attacker of the LAA is possible to attack the memory space of another process when it is running, and the malicious attacking process must exist in its own memory space or embed in the memory space of other processes. When the malicious process or its embedded process is loaded into memory space, the model can detect it at the beginning of the execution of the malicious program.

(3) *Security Analysis.* The security of the MP measurement model is supported by the security of the measurement root and measurement algorithm.

The TPM is a measurement root of the MP measurement model. In this model, both integrity calculation and integrity verification execute TPM and rely on the security of TPM. In general, the TPM module is a secure chip, and it regards internal cryptographic algorithm engine as a credible guarantee of the trusted computing. As a measurement root, the TPM module is the first node of trusted chain in the trusted computing platform, and the rest of the trusted computing will regard this step as a foundation. The security of TPM is mainly reflected in the ability of chip design security, key storage, antisoftware analysis, and antihardware analysis. Hence, the security of the TPM is ensured by physical security and administrative security.

This model uses hash algorithm as a measurement algorithm. It is important to select a reliable hash function to ensure credibility of this measurement mechanism.

(4) *Computational Complexity Analysis.* This model uses hash algorithm as a measurement algorithm. The computational complexity of the MP model depends on the hash algorithm.

(5) *Comparative Analysis with Other Measurement Models.* In Table 1, several integrity measurement methods are compared in four aspects: range of action, hardware requirement, structure complexity, and realization mechanism.

The TCG measurement model can only accomplish the static integrity measurement in the platform starting phase. Because this measurement method is only executing integrity


```

//xen/include/public/xen.h
struct mmuext_op{
    unsigned int cmd;
    union{
        xen_pfn_t mfn;
        unsigned long linear_addr;
    } arg1;
    union{
        unsigned int nr_ents;
        Xen_GUEST_HANDLE_00030205(void) vcpumask;
    } arg2;
};
typedef struct mmuext_op mmuext_op_t;
DEFINE_Xen_GUEST_HANDLE(mmuext_op_t);

```

ALGORITHM 1

TABLE 1: Comparative analysis with other measurement models.

	Range of action	Hardware requirement	Structure complexity	Realization mechanism
TCG	Platform starts	TPM	Simple	Trusted chain
IMA	Program load	TPM	Simple	LSM
SB	Program run	TPM + LT/SEM	Complex	Process isolation
MP	Program run	TPM	Simple	Virtual machine

measurement for the component loaded to platform, it is easy to realize.

The IMA measurement model expands the TCG measurement model and executes integrity measurement at the program load time. However, due to the limitation of the LSM mechanism, the integrity measurement cannot be implemented when the program is running.

By using the process isolation, the SB measurement model can implement dynamic integrity measurement when the program is running. But it needs special hardware support. LaGrande Technology (LT) is a hardware technology introduced by Intel, and it aims at the Palladium trusted computing program of Microsoft. Secure Execution Mode (SEM) is a hardware technology proposed by AMD. In the SB, and TPM, LT and SEM are used for isolating memory from the software attack in the running.

The MP model can implement dynamic integrity measurement when the program is running. Moreover, its structure is simple and easy to be realized. The rest of this paper will discuss how to realize the MP integrity measurement model under XEN virtual architecture. This realization mechanism is simple and flexible, and it also avoids the limited process communication issues because of the process isolation.

3. Realization of MP Based on XEN Super Call

3.1. Realization Mechanism. In XEN system, as a virtual monitor, XEN is located in the highest privilege level (ring 0), while Guest OS is only located in the sublevel (ring 1). The lower kernel privilege makes Guest OS not execute some privileged operations, and XEN can execute privileged operations for it. Therefore, XEN provides a series of interface, and it is convenient for Guest OS to complete those privileged operations, called super call [14].

When the executable subject is in operation, Guest OS can manage and handle the memory space based on characteristics of super call mechanism. If page table operations are involved, a super call to the XEN begins. At this time, we insert a measurement point and embed measurement program in the processing of super call function. And then this model can be realized. The specific process is shown in Figure 3.

- (1) Guest OS loads page into memory and tries to handle the page table.
- (2) Guest OS cannot access the page table directly, and it applies to XEN super call. The XEN checks the super call request, and it will go to the next step if it is permitted, otherwise the program ends.
- (3) Implement measurement program: the measurement program gets the memory page, calls TPM hash algorithm for calculating measurement value, and compares it with the reference value for the integrity of these pages. If they are consistent, it means that the page is trusted, and the program goes to the next step, otherwise it ends.
- (4) Complete the operation on the page table and return to Guest OS.

3.2. Realization of the Measurement Model

(1) Measure When Creating Page Table. When an executable subject carries out, this software creates a complete copy

FIGURE 3: Measurement flow process diagram based on XEN hypercalls.

```

//xen/include/public/xen.h
#define MMU_NORMAL_PT_UPDATE    0
#define MMU_MACHPHYS_UPDATE    1

```

ALGORITHM 2

```

struct mmu_update {
    uint64_t ptr;    /* Machine address of PTE. */
    uint64_t val;    /* New contents of PTE. */
};
typedef struct mmu_update mmu_update_t;
DEFINE_XEN_GUEST_HANDLE(mmu_update_t);

```

ALGORITHM 3

of program map on the hard disk, and the address space can come into being. But the software is entirely not put into memory to handle. Guest OS just allocates a number of memories and creates page table to map physical memory and address space. Guest OS calls `HYPervisor_mmuext_op` and loads this process page table into page directory. After XEN finishes the authority inspection, it executes the `do_mmuext_op()` function. At this time, measurement program inserts measurement point into this function and executes the measurement program.

XEN version-3.4.0 sustains 18 kinds of operation about `HYPervisor_mmuext_op`, including mounting the page table, unloading page table, updating CR3 register, cache refresh, and LDT table settings. The `HYPervisor_mmuext_op` operation shares one parameter structure `mmuext_op` in Algorithm 1.

The structure body contains two unions: `arg1` and `arg2` and contains four parameters `mfn`, `linear_addr`, `nr_ents`, and `vcpumask`. The definition of the structure body changes according to different operation. In the page table operation, `mfn` points to machine address of the page table.

The measurement program exactly gets the page according to the parameters.

(2) *Measure When Updating Page Table.* When one program runs, all the dynamic change will reflect in the memory, and the change of memory will reflect in the page table. When the process changes, the memory changes, and the page table also changes. Guest OS applies for using `_HYPervisor_mmu_update` to update the page table. After XEN finishes the authority inspection, it executes the `do_mmu_update()` function. At this time, the measurement program inserts measurement point into this function and executes integrity measurement.

There are two kinds of operation defined in the `_HYPervisor_mmu_update`:

page table update (`MMU_NORMAL_PT_UPDATE`)

M2P table update (`MMU_MACHPHYS.UPDATE`),

which are defined in Algorithm 2.

The structure `mmu_update` defined in `_HYPervisor_mmu_update` includes two parameters: "ptr" and "val". The ptr stands for the physical address of the item which needs to update the page table, while the val stands for the new value which needs to change the page table. The structure is defined in Algorithm 3.

In Guest OS, the page table update means the item update of page table. What the item of page table saves is actually a memory page address (or page frame number). Therefore, in fact, the item update of page table is equivalent to mapping a new memory to the current item of page table to replace the original page. In this process, both sides of operation are the original page address and new mapping page address, and it is not (ptr, val) but (val, new val). Thus, XEN defines a new hypercall to finish this kind of page table update, which is `_HYPervisor_update_va_mapping`. Because this kind of operation is relatively simple, it does not have the definition structure as other hypercalls, and it transfers parameters directly. Under x86 framework, the three parameters needed in the operation are transferred through register EBX, ECX, and EDX to `do_update_va_mapping()`.

And then the measurement program inserts measurement point into this function and executes integrity measurement.

(3) *Writable Page Table Measurement.* The Hypercall method makes the update page table model become indirect; that is, Guest OS needs to get XEN safety confirm and use XEN for completing the page table update. Thus, Guest OS is unable to write its page table. But, in fact, other than using the hypercall way to update the page table, XEN also provides a new page table update model named writable page model. Though the page is still unable to write in this model, Guest OS can read-write these kinds of pages directly.

Of course, in the writable page model, Guest OS does not really have the access to write page table. After finishing the update operation of the page table, it still needs XEN to confirm the operation. Before Guest OS updates the page table, XEN takes the page table from the page table structure of Guest OS and makes Guest OS recognize the page as ordinary page to read and write. After Guest OS finishes the update and XEN confirms it, XEN will put the page into the page table structure of Guest OS again and allow the MMU to access this page table.

In the writable page model, Guest OS can update the page table by using Hypercall: `_HYPERVISOR_vm_assist`; the function is `vm_assist()`. And then, the measurement program inserts measurement point into this function and executes integrity measurement.

4. Conclusion

In order to solve the existed problems of dynamic integrity measurement method, a dynamic integrity measurement model based on Memory Paging Mechanism is proposed in this paper. The model takes executable subject pages as measurement object, inserts measurement points, and measures and verifies them before the pages are loaded into memory. This model insures the integrity and trust of each executable page, and the integrity of the whole executable subjects is not destroyed. To prove this model, XEN hypercall mechanism is used for acquiring executable subject pages loaded into memory, putting measurement codes and verification codes into hypercall handler, and then the dynamic integrity measurement to executable subjects is realized.

In the next work, memory attack model will be discussed further, and the integrity measurement algorithm will be improved on the same steps.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] C. Song, W.-P. Peng, Y. Xin, S.-S. Luo, and H.-L. Zhu, "Seal-based secure boot scheme for trusted computing platform," *The Journal of China Universities of Posts and Telecommunications*, vol. 17, supplement 2, pp. 16–21, 2010.
- [2] A. Nagarajan and V. Varadharajan, "Dynamic trust enhanced security model for trusted platform based services," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 564–573, 2011.
- [3] J. Winter and K. Dietrich, "A hijacker's guide to communication interfaces of the trusted platform module," *Computers & Mathematics with Applications*, vol. 65, no. 5, pp. 748–761, 2013.
- [4] A. K. Kanuparthi, M. Zahran, and R. Karri, "Architecture support for dynamic integrity checking," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 321–332, 2012.
- [5] D. Muthukumaran, J. Schiffman, M. Hassan, A. Sawani, V. Rao, and T. Jaeger, "Protecting the integrity of trusted applications in mobile phone systems," *Security and Communication Networks*, vol. 4, no. 6, pp. 633–650, 2011.
- [6] D. Schellekens, B. Wyseur, and B. Preneel, "Remote attestation on legacy operating systems with trusted platform modules," *Electronic Notes in Theoretical Computer Science*, vol. 197, no. 1, pp. 59–72, 2008.
- [7] D.-F. Li, Y.-X. Yang, L.-Z. Gu, and B. Sun, "Study on dynamic trust metric of trusted network based on state and behavior associated," *Journal on Communications*, vol. 31, no. 12, pp. 12–19, 2010.
- [8] R. Sailer, X. L. Zhang, T. Jaeger, and V. Doom, "Design and implementation of a TCG based integrity measurement architecture," in *Proceedings of the 13th USENIX Security Symposium*, pp. 223–238, 2004.
- [9] T. Jaeger, R. Sailer, and U. Shankar, "Prima: policy reduced integrity measurement architecture," in *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT '06)*, pp. 19–28, New York, NY, USA, 2006.
- [10] X. Zhang, M. J. Covington, S. Q. Chen, and R. Sandhu, "Secure-Bus: towards application-transparent trusted computing with mandatory access control," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS '07)*, pp. 117–126, New York, NY, USA, 2007.
- [11] L. Gao, X. Qin, C. Chang, and X. Chen, "A embedded system-based computing platform for tolerating untrusted component," *Geomatics and Information Science of Wuhan University*, vol. 35, no. 5, pp. 626–629, 2010.
- [12] S. Stamm, N. P. Sheppard, and R. Safavi-Naini, "Implementing trusted terminals with a and SITDRM," *Electronic Notes in Theoretical Computer Science*, vol. 197, no. 1, pp. 73–85, 2008.
- [13] M. Thober, J. A. Pendergrass, and C. D. Mcdonell, "Improving coherency of runtime integrity measurement," in *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (STC '08)*, pp. 51–60, New York, NY, USA, 2008.
- [14] S. Grinberg and S. Weiss, "Architectural virtualization extensions: a systems perspective," *Computer Science Review*, vol. 6, no. 5-6, pp. 209–224, 2012.