

Integrity Management Infrastructure for Trusted Computing

Seiji MUNETOH^{†a)}, Megumi NAKAMURA[†], Sachiko YOSHIHAMA[†], *Nonmembers*,
and Michiharu KUDO[†], *Member*

SUMMARY Computer security concerns have been rapidly increasing because of repeated security breaches and leakages of sensitive personal information. Such security breaches are mainly caused by an inappropriate management of the PCs, so maintaining integrity of the platform configuration is essential, and, verifying the integrity of the computer platform and software becomes more significant. To address these problems, the Trusted Computing Group (TCG) has developed various specifications that are used to measure the integrity of the platform based on hardware trust. In the trusted computing technology, the integrity data of each component running on the platform is recorded in the security chip and they are securely checked by a remote attestation. The infrastructure working group in the TCG is trying to define an Integrity Management Infrastructure in which the Platform Trust Services (PTS) is a new key component which deals with an Integrity Report. When we use the PTS in the target platform, it is a service component that collects and measures the runtime integrity of the target platform in a secure way. The PTS can also be used to validate the Integrity Reports. We introduce the notion of the Platform Validation Authority, a trusted third party, which verifies the composition of the integrity measurement of the target platform in the Integrity Reports. The Platform Validation Authority complements the role of the current Certificate Authority in the Public Key Infrastructure which attests to the integrity of the user identity as well as to related artifacts such as digital signatures. In this paper, we cover the research topics in this new area, the relevant technologies and open issues of the trusted computing, and the detail of our PTS implementation.

key words: *trusted computing, integrity management, vulnerability management*

1. Introduction

Currently, security incidents are frequently reported, such as information leakage of sensitive personal data, swindles based on phishing sites, password leakage by key-loggers, zero-day attacks, to name a few. A primary root cause is malware, involving the installation of malicious software which is not recognized by users as being malicious, according to the CSI/FBI Computer Crime and Security Survey 2006 [1]. For example, an unsuspecting user may invoke an executable file attached to email sent by an attacker, which installs a malicious key-logger into the operating system, and which then stealthily transmits any typed passwords to a remote site. Thus such cunning malware sneaks valuable information out of an enterprise's network firewall and intrusion detection systems once considered to be effective enough to protect the IT environment.

To prevent such incidents from happening, various

kinds of security software are available these days, ranging from intrusion detection systems to virus detection tools that also protect a user's computing platform from malicious attacks by malware such as BotNet installers. However, security tools based on software-rooted trust are sometimes not effective against rootkit and self-evolving malware that hide themselves and can sneak through the end-point protective mechanisms provided by the security tools. For example, a metamorphic viruse which modifies and encrypts virus code when copying itself is not detectable by protection mechanisms using blacklist pattern matching schemes. These problems are becoming more and more critical since end-user devices such as client PCs, PDAs, and smart phones are playing important roles in the handling and storage of sensitive business data. The security survey also reports that virus attacks continue to be the source of the main financial losses, which amounted to 16 million dollars in 2006 just among the survey's respondents.

Public Key Infrastructure (PKI) technology provides the SSL/TLS authentication scheme which is used to check the authenticity of the server and client platform over the wire. However, the SSL authentication does not necessarily mean that the endpoint platform is uncontaminated and correctly configured as the users expect. The SSL certificate only claims that the public key pair associated with the session is certified by a certain Certificate Authority (CA) or Registration Authority (RA), which implies the authenticity of the organization (or the owner of the platform) but has nothing to do with the authenticity or the correctness of the endpoint platform.

The Trusted Computing Group (TCG) [2] focuses on these problems using a completely different and innovative new approach. The TCG's trust model is hardware-rooted, based on the existence of a specific security chip embedded in the computing platform, and uses a whitelist-based software component verification scheme. The TCG's trusted computing technology makes it possible to check the health of a computing platform (using a technique called Remote Attestation) in a more secure and trusted manner and to alert the user whenever the platform is contaminated by unknown software. Using TCG's Remote Attestation, the integrity data for each component running on the platform is recorded in the security chip and securely checked by remote attestation. The infrastructure working group in the TCG is defining an Integrity Management Infrastructure in which the Platform Trust Services (PTS) is a key new com-

Manuscript received November 27, 2007.

[†]The authors are with Tokyo Research Laboratory, IBM Japan, Yamato-shi, 242-8502 Japan.

a) E-mail: munetoh@jp.ibm.com

DOI: 10.1093/ietisy/e91-d.5.1242

ponent which deals with the integrity information of a platform, but the realistic PTS architecture has not been defined due to the complexity and variety of the runtime platforms.

In this paper, we introduce the concept of the Platform Validation Authority (PVA), a trusted third party which verifies the composition of each target platform reported in remote attestations through the use of the PTS. The PVA complements the role of the current CA in the PKI by adding a new feature of platform attestation. We proved technical feasibility of the PVA and the PTS by proposing novel schemas for solving open issues in platform integrity validation.

The rest of this paper is organized as follows. Section 2 reviews related work, and then Sect. 3 introduces trusted computing and identity management infrastructure. Section 4 presents the design and implementation of our PTS, Sect. 5 shows same experimental results for our prototype. Section 6 concludes the paper.

2. Related Work

The Aegis system [3] provides secure bootstrapping architecture on PC systems that maintain an integrity chain from the lowest trustable layer of a system. Trusted bootstrap is different from the secure bootstrap, it continuing to bootstrap even when the integrity of the system is compromised, and defers any decision until when the attestation of platform integrity is challenged.

Several prior projects have introduced use cases for trusted computing. Sailer et al. [4] uses the integrity measurements and attestations to protect remote access points. The Trusted Network Connect (TNC) specification [5] of the TCG defines a generic network authentication architecture based on the platform configuration state. The TNC supports an option to use TPM to measure and report the platform integrity using remote attestation. Terra [6] provides isolated trusted platforms on top of a virtual machine monitor, and allows attestation by a binary image of each virtual machine, such as virtual disk images, virtual BIOS, and VM descriptions. The binary attestation approach of Terra is suitable for closed-architecture platforms such as ATMs and game boxes. Trusted Virtual Domains [7], [8] proposes a framework for establishing trusted coalitions based on the platform integrity information.

The verification of binary attestation has been recognized as a difficult research problem for years. Halder et al. proposed semantic remote attestation [9], in which an agent on the virtual machine (i.e., the Java VM) attests to the platform's state with higher-level properties that express the behavior of the portal code, rather than low-level integrity measurements in SHA-1 hash values. Property-based Attestation [10], [11] proposes an attestation model with a trusted third party that translates low-level integrity information into a set of properties.

WS-Attestation [12] proposes an attestation protocol mapping on Web Services protocol, and allows a trusted third party to issue a security token as a result of the ver-

ification. It also proposes an initial idea for a vulnerability database used for platform integrity verification, which has been substantially extended in this paper.

However, it has been an open issue how to verify the low-level integrity information of complicated computing platforms. Our verification mechanism allows more flexibility and adaptability in verifying the security of PC platforms that have a larger number of varieties of running software components.

3. Trusted Computing and Infrastructure

This section introduces the basic architecture of Trusted Computing as well as the two main aspects of the supporting infrastructure: *identity management* and *integrity management*.

3.1 TCG Security Architecture

The TCG [2] defines a set of industry-standard specifications for hardware and software to enable trusted computing among a wide variety of computing platforms, such as PCs, servers, mobile phones, etc. The center of the TCG technology is a security module called the Trusted Platform Module (TPM), which is usually implemented as a tamper-resistant hardware module. In addition to serving as a cryptographic co-processor and protected storage for secrets and keys, the TPM is used to measure and report platform integrity in a manner that cannot be compromised even by the platform owners or the software running on it. In this section, we review the mechanisms of platform integrity measurement and platform integrity reporting (i.e., attestation) in more detail.

3.1.1 Platform Integrity Measurement

Platform integrity measurement consists of multiple phases of measuring and storing the fingerprints of the hardware and software components that make up a platform. Integrity measurement can be categorized into two types: 1) the TCG trusted bootstrap and 2) other integrity measurements built on top of the trusted bootstrap.

TCG defines the trusted bootstrap process [13], that comprises an iterative process of "measurement", loading, and execution of software components. When the system is powered-on, the immutable initial bootstrap code (such as the BIOS boot block) measures the next component and stores the measurement in the TPM before transferring control to the next component. In subsequent steps, each software component recursively measures the next component and records the measurements in the TPM, until the operating system is loaded. The BIOS boot block and the TPM are called the Core Root of Trust for Measurement (CRTM), because they need to be trusted in order to trust the measurements in the trusted bootstrap.

Each measurement uses a SHA-1 hash value of the binary image of a component, and stores it into a Platform Configuration Register (PCR). PCRs are special purpose

registers within the TPM which record the integrity measurements, and which are protected from arbitrary modification.

The specification requires a minimum of 24 PCRs on a TPM, but each PCR can store the fingerprints of multiple components using a hash-chain mechanism. That is, a PCR supports only an *extend* operation to update its value; i.e., When recording a measurement value m into a PCR, the value m is extended into the PCR, which results in a SHA-1 hash over the concatenation of the current PCR value and the value m .

$$PCR_i^n = SHA1(PCR_{i-1}^n || m)$$

where the initial PCR value after power-on is $PCR_0^n = 0$, n denotes the index of the PCR register, and $||$ denotes a concatenation.

After the OS is loaded and initialized in a trusted bootstrap process, the PCRs will contain some predictable values, provided that the platform runs a known set of components and they are intact. If any bits of the measured components are modified from the originals, such modification can be detected by checking the PCR values.

The trusted bootstrap provides a basis for platform integrity measurement, but it does not prove all aspects of trustworthiness of a system. Various modules and application programs that are loaded after the bootstrap process contribute to the trustworthiness of the platform. Similarly, even identical OS kernels can have different levels of security if they have different configuration settings, therefore such configuration information needs to be measured. The Integrity Measurement Architecture [14] supports such measurements for the Linux[†] kernel, and enhances the role of the TPM for measuring not only the static state of a system but also the dynamic state. Several proposals have been made [9], [10], [12] to measure additional states, including the configuration of the OS, middleware, and applications, to further enhance the role of TPM to the upper layers of the software stack. Care needs to be taken, though, that a transitive trust chain is maintained from the CRTM all the way to the component being measured.

The TPM specification version 1.2 introduced a new mechanism to support virtual machines, by resetting the state of PCRs in a special mode, in which the CPU transfers its control to a secure kernel that hosts virtual machines. With a special instruction in the CPU, the TPM resets PCR16-23, and then measures integrity of the secure kernel using PCR17. This mechanism is called the Dynamic Root of Trust for Measurement (DRTM), in contrast to the Static Root of Trust for Measurement (SRTM) in the trusted bootstrap process. An advantage of the DRTM is that it does not need to trust the BIOS as the CRTM, and removes it from the transitive trust chain. In the DRTM, only the trust chain starting from the secure kernel needs to be maintained.

3.1.2 Platform Integrity Reporting

Platform Integrity Reporting, or remote attestation, is a

mechanism defined in the TCG specification [13] to report integrity measurements stored in the PCRs in a reliable manner. In the attestation process, the TPM signs the PCR values and an external 160-bit challenge (such as a nonce from a challenger to avoid replay attacks) using an RSA private key, the confidentiality of which is protected by the TPM. The attestation is an atomic, protected operation and the attestation signature cannot be forged by malicious software. Therefore, if the TPM is properly designed and implemented to adhere to the TPM specifications, and the platform, including the initial bootstrap code, is properly integrated with the TPM, a remote verifier can have confidence in the integrity measurements reported by the TPM.

The platform may also send additional information in an attestation, such as the integrity measurement log, which includes a list of components loaded on the platform and measured by the TPM, and how these components are composed. Realistically, verification of a PCR value is quite difficult without such supporting information, since each PCR holds a hash-chain of measurement of multiple components, and the number of permutations of the hash-chain could be exponential. The integrity of the measurement log can easily be verified by checking it against the PCR values.

3.2 Identity Management

TCG defines several types of keys and credentials to prove the genuineness of each TPM while protecting its privacy.

EK and EK Credential: An Endorsement Key (EK) is an RSA key unique to each TPM instance. Typically the EK is generated during the TPM manufacturing process, and the private key is protected inside the TPM, and the TPM manufacturer issues an EK credential to assert the genuineness of the TPM. However, because of privacy concerns, the EK is not used directly for attestation or other purposes, but its use is limited to proving the genuineness of TPM in the AIK credential issuance process.

Platform Credential: Typically a platform credential is issued by a platform manufacturer (e.g., PC vendor), to assert that the design and implementation of the computing platform conforms to the TCG specification, e.g., the trusted bootstrap process is implemented as specified. The platform credential associates an EK credential with a platform.

AIK and AIK Credential: An Attestation Identity Key (AIK) is an RSA key that is used only for the attestation. Each TPM may create and own an arbitrary number of AIKs. Each AIK credential asserts that the key belongs to a genuine TPM, without revealing which instance of the TPM.

A TCG defines a special Certification Authority called a Privacy CA which issues the certificates for AIKs. In the

[†]Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

certification process, the requesting TPM first generates a pair of AIK, and send its public portion along with the EK credential and the platform credential to the privacy CA. The Privacy CA is responsible for verifying these credentials to make sure that the request comes from a genuine platform with a genuine TPM. The Privacy CA returns the AIK certificate encrypted with the EK public key, and thus only the requesting TPM with the genuine EK can decrypt it. The Privacy CA, as its name indicates, is also responsible for protecting the privacy of the TPM; i.e., the connection between the EK and the AIK must be kept secret by the Privacy CA.

3.3 Integrity Management

Integrity is a core concept in Trusted Computing. The term “integrity” is used to denote the state of the components within a computing platform, and the ultimate objective of Trusted Computing is to ascertain that the components in a computing platform are in known and trustworthy states, and have not been tampered with. (Note, however, that the TCG’s notion of integrity does not guarantee the correctness of implementations.)

The TCG specification defines the Integrity Management Model (IMM) [15], to create an infrastructure for managing component integrity. The IMM, together with the integrity measurement and reporting capability of TPM, allows verification of the integrity of the components in a computing platform.

In the TCG specifications, the term “Integrity Measurement” has two meanings. The first is the runtime integrity measurements on a computing platform with TPM. The second is the reference measurements by platform manufacturers. The assertion made by a manufacturer on the composition of components and their integrity metrics (i.e., hash values) is called the Reference Manifest (RM). The RM is later compared with the runtime measurements to verify the integrity of the platform configuration.

Typically, remote attestation is used in a protocol where a requester (i.e., a computing platform with TPM) is seeking some interaction with a relying party (e.g., a service or an access to some resources), and the relying party needs to trust not only (or not necessarily) the identity of the requester but the integrity of the computing platform. The requester sends a platform integrity report to the relying party. The relying party may act as a verifier, or asks a trusted verifier to verify the report on its behalf. In either case, since the verifier may not be in the same trust domain as the requester, it may not know all of the software and hardware components that constitute the requesting computing platform. The Reference Manifest is used by the verifier as the authoritative information to evaluate the integrity report.

The TCG IMM defines several steps in the lifecycle of integrity measurement.

Creation: Integrity metrics of components are generated by hardware manufacturers and software vendors, and

the Reference Manifests for these components are issued.

Collection: When components are gathered to constitute a computing platform, corresponding integrity information is collected. E.g., this step is done by the IT department of a corporation that is responsible for configuration of client PCs in the enterprise.

Communication/Publication: Collected integrity information is made available to the verifier, e.g., through a public repository of RMs.

Collation: When the verifier seeks to verify an integrity report, the verifier collects the RMs in order to use it for evaluation. This step is called collation.

Evaluation: The verifier evaluates the integrity report against the RMs, as well as additional information (e.g., the latest vulnerability information) if such information is available.

The PTS is a standard component defined in the IMM that provides necessary services such as measuring components, parsing RMs, interacting with the TPM and creating integrity reports. The applications or an integrity measurement collector (IMC) interact with the PTS.

3.4 Platform Trust Services (PTS)

The IMM defines the PTS as a standard IMM component which provides all the necessary services for integrity measurement. The TCG specification defines both the client-side PTS and the server-side PTS. Figure 1 shows overall PTS Client/Server architecture.

The client-side PTS is an entity which generates Integrity Report (IR), by collecting information from the Integrity Measurement Log (IML) and the PCR values signed by the TPM with an AIK, through TCG Software Stack (TSS).

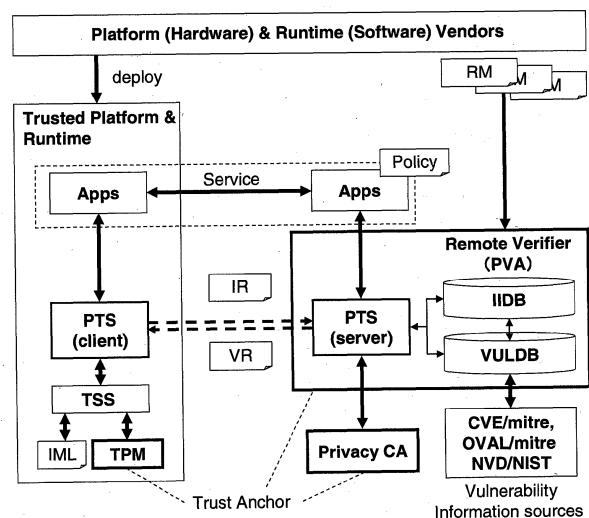


Fig. 1 PTS Client/Server architecture.

The server-side PTS is an entity which verifies the IRs and returns a Verification Result (VR). The result is one of three values: *valid*, *invalid*, or *unverified*, which means that the IR is verified as trusted, the IR is verified to be invalid, or there is not enough information for making a decision, respectively. The server-side PTS verifies IRs against the RM. The integrity of the RM, which represents the expected platform integrity metrics, is protected by the signature by the issuer, e.g., the platform vendor or a PVA.

The vendors release software with RMs and the software is deployed into the target platforms. The PVA maintains the integrity information database (IIDB) and the vulnerability database (VULDB) for improving performance of validation. The IIDB keeps digests and package information obtained from RMs. The latest vulnerability information are retrieved from external sources, such as CVE [16], OVAL [17] and NVD [18].

The PTS allows applications running on the platform to be measured and attested before it starts using the service, and thus allowing to establish trust between the client and the server.

The Integrity Measurement Collector (IMC, which fills the role of the client-side 'Demo App' in Fig. 3) [5] or other measurement controlling agents can communicate with the PTS to parse the RMs, create snapshots of the platform integrity, synchronize the snapshots, and generate IRs.

The PTS is the primary IMM component that is responsible for interacting with the TPM when this trusted hardware is available on the client platform. The capabilities of the PTS that are available to an IMC or other measurement controlling agents are defined in the TCG IF-PTS specification [19].

The standard formats managed by the PTS are defined using an XML Schema in the TCG specifications as follows:

Reference Manifest Schema is the format for communication of the platform Integrity Information. The RM is signed and issued by a platform or software vendor, and is used by the verifier (i.e., PTS) as the authoritative information.

Integrity Report Schema represents a snapshot of the platform integrity measurement, including the IML, and signed by the attestation signature of the TPM to attest to the authenticity of the report. The IR may also include assertions made by the components running on the client platform.

Validation Result Schema communicates a verification result from the verifier (e.g., the server-side PTS) to the client. The result is one of the following: VALID, INVALID, or UNVERIFIED. Also it contains Reason-Strings for a detailed reason when the result is not VALID. To validate an IR to derive a result, the PVA and the relying party need to share a policy that describes the detailed conditions of the target platform. The issue will be discussed in Sect. 4.3.

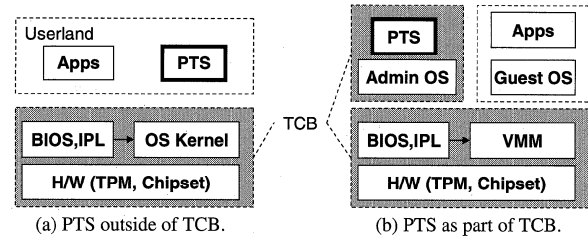


Fig. 2 Two type of PTS implementation.

3.4.1 PTS Architecture

Figure 2 shows two different approaches for the implementation of the client-side PTS. Figure 2(a) shows the PTS implementation in our prototype on ordinary Linux. In (a), the PTS runs as an application in the user space, and thus is not part of the trusted computing base (TCB). In this case, the TCB has an integrity measurement function and needs to cooperate with the PTS, and userland applications, include the PTS, are measured by the OS Kernel.

Figure 2(b) shows the PTS implemented on top of virtual machine monitor (VMM) such as the Xen Hypervisor [20] or the Intel Virtual Appliance [21]. The PTS runs in the admin VM on the VMM, and thus is part of the TCB. The PTS can then issue proving commands to measure the integrity of the other guest OSes. The design of Fig. 2(b) is intended by the PTS specification, since it allows measurements and the assertion of platform integrity in a trustworthy fashion. However, the approach in Fig. 2(b) calls for a VMM, and an efficient measurement technology against the Guest OS remain to be solved.

Figure 2(a) is a lightweight solution that runs on the existing PC platforms without needing to run a VMM. Since in most of the legacy operating systems, the TSS is implemented as a userland service, and the PTS runs on top of the TSS, we chose to implement the PTS as a user space application. This makes sense because both the TSS and the PTS are too complicated to be implemented as part of the TCB. This type of PTS implementation gathers integrity information from the TCB and generates a report with a TPM attestation signature.

The Integrity Measurement Architecture (IMA) [14] can be used for this and the integrity of the PTS is also protected by the transitive trust chain. An attacker may modify the PTS but the verifier can detect such an alteration, since the completeness of the generated IR does not depend on the PTS itself.

4. Implementation of Open Platform Trust Services

In this section, we expound on our prototype, the Open Platform Trust Services [22]. Our contribution for solving open issues in platform integrity validation includes 1) a state-machine based model for validation of integrity reports which enables verification of the transitive trust chain

of a platform runtime, 2) the integrity database which integrates the expected integrity metrics of software packages with vulnerability information obtained from external security advisories, 3) identified a set of vocabularies for expressing required properties of target platforms.

The prototype is implemented in the Java language and the iBatis/JDBC library is used, allowing database access for validation of IRs with the latest vulnerability information. We also provide a sample application for remote attestation, where HTTPS is used for the client/server communications by Apache Tomcat.

We use the same PTS component for the client and the server as shown in Fig. 3. Thus the client-side PTS may also verify the IR against the RM. The verification by the client-side PTS filters out obviously erroneous IRs without needing to send them to the PTS server. Since each IR can become very large (e.g., 100 KB or more), the client-side verification reduces the unnecessary traffic to the server.

The server-side PTS may be handled by various entities depending on the use-case of the remote attestation. The PVA shown in Fig. 3 manages *Collection* through *Evaluation* steps in the TCG IMM. The collected RMs are stored in the IIDB for efficient verification. The IIDB maintains a whitelist (or blacklist) of hash values of the known components. In addition, each entry of the IIDB has a flag that indicates if there are any known vulnerabilities in the component. The VULDB is the central repository of the known vulnerabilities, by aggregating information from vendors and security advisories. The VULDB synchronizes with the IIDB to update the IIDB with the latest vulnerability information. Separation of the roles allows the IIDB to avoid the complexities of vulnerability management and remain able to quickly respond to the queries from the PTS.

The communications between the client-side and the server-side PTS are independent of the application-layer protocol. In our prototype, the IR and the verification result (both in XML) are base-64 encoded and exchanged in the HTTPS message body. However, other choices, such as use of the WS-Trust protocol as proposed in [12] are also possible. The Trusted Network Connect (TNC) [5] is a realization of platform attestation integrated with network authentication, in which case the 802.1x supplicant and the authentication server respectively become the ‘Demo App’ and ‘Tomcat App’ in Fig. 3.

The client-side PTS and demo applications are writ-

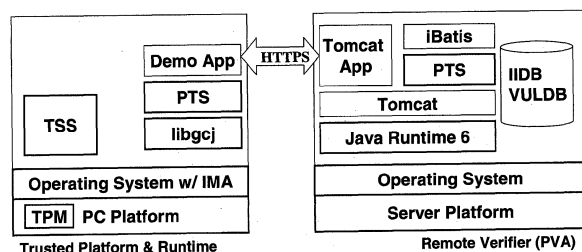


Fig. 3 Our prototype implementation of PTS and demo app.

ten in Java and compiled into native code using GCJ (GNU Compiler for Java) since IMA cannot measure Java classes running under the Java runtime. With this transformation, all relevant executables can be measured and validated by PVA.

4.1 Integrity Measurement and Validation with FSM

The current TCG specification defines only an abstract concept of the PTS [19]. However in reality, the PTS needs to have knowledge about the platform architecture, including the composition of various components running on the platform.

Figure 4 shows the data flow of our PTS implementation corresponding to the TCG IMM, which includes some extensions to the current standard specifications.

First, the PTS can generate Reference Manifests (RMs) and Integrity Reports (IRs) for the platform based on the given behavior model and the IML.

Second, we have extended the format of the RM to incorporate the behavior model, allowing the PTS to use it when verifying the IRs. Intuitively, the behavior model is a finite state machine (FSM) that can consume an IML. The PCR values and the IML in the IR are generated as a result of running the trusted platform, while the behavior model describes how they are measured, and how they should be verified.

The FSM is a representation of the trusted-bootstrap and subsequent measurement processes of the platform. Each state in the behavior model represents a platform state, and each transition represents the integrity of the component that is loaded and executed. This extended capability of the PTS supports efficient verification and management of the integrity of the platform.

We use an Unified Modeling Language version 2 (UML2) state diagram to represent each FSM, and its XML representation is included in the extensible part of the RM. Figure 5 shows a brief example of UML2 which represents the transitive trust chain. In this example, the state diagram is driven by a given IML. The IML format, usage of

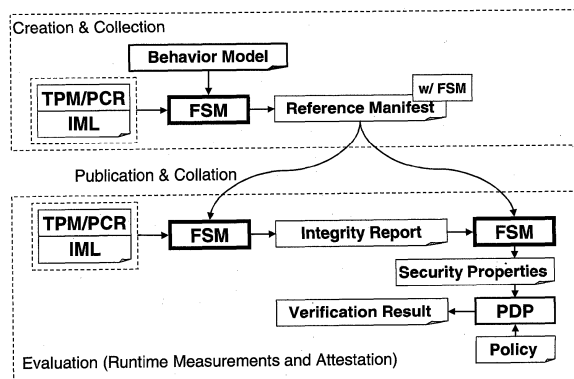


Fig. 4 IMM steps and data flow with FSM.

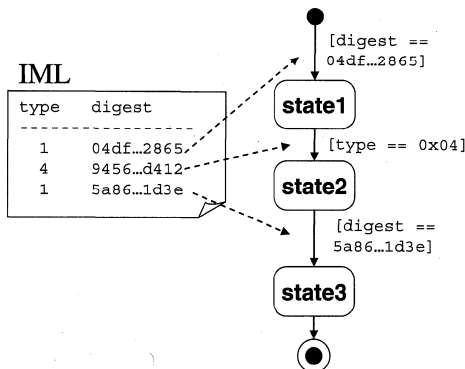


Fig. 5 FSM representation of transitive trust chain.

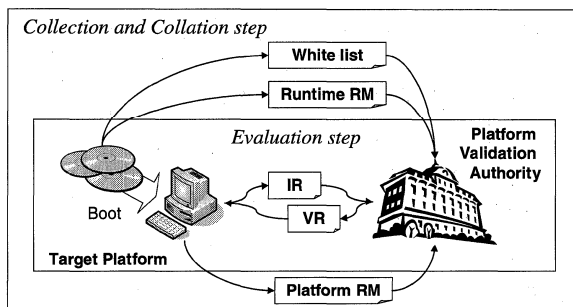


Fig. 6 IMM for CD bootable O/S.

the PCRs and events are defined by the TCG PC specification [23]. Each event contains a PCR index, digest, an event type and the event data.

In this example, the digest must be 04df ... 2865 and 5a86 ... 1d3e to make the transition from the initial state to state1, and state2 to state3, respectively. For the transition from state1 to state2, only the event type 0x04 is required. Therefore, the IML in Fig. 5 passes the FSM.

The FSM can be packaged at various levels of granularity. Here, the verification of a platform is possible using FSMs for an arbitrary combination. In the example of Fig. 6, the set of two FSMs, Platform (BIOS), and Runtime (O/S), describes the platform structure.

4.2 Database Architecture

Our server-side PTS implementation contains the integrity database which integrates the expected integrity metrics of software packages with vulnerability information obtained from external security advisories. Since our target platform supports Linux-IMA and the IR includes many of vulnerable userland components which need to be checked with the latest status.

4.2.1 Vulnerability Management

Information security vulnerabilities can be identified by

CVE (Common Vulnerabilities and Exposures) Identifiers [16] managed and provided by MITRE Corporation. CVEs are unique identifiers used widely, and make it easier to share data across different tools provided by different vendors. Security advisories such as RSHA (Red Hat Security Advisory) [24] for Red Hat packages and DSA (Debian Security Advisory) [25] for Debian packages include the CVE identifiers in the vulnerability information.

The CVSS (Common Vulnerability Scoring System) [26] provides a method for rating vulnerability scores in an open and universal manner. This method evaluates vulnerabilities by three metrics such as base metrics, temporal metrics and environmental metrics. The NVD (National Vulnerability Database) [18] provides CVSS scores for most of the CVE entries, and thus each security advisory can be associated with a vulnerability score.

We inspect platforms with reference to these vulnerability information sources. Although there are vendor specific vulnerability checking tools, we use OVAL (Open Vulnerability and Assessment Language) [17], which is a generic open framework for representing the vulnerability test processes. OVAL can describe the test cases to detect the presence of vulnerabilities. From the package management perspective, OVAL describes the versions of the base software distributions and how to check the vulnerable component versions. In addition, OVAL can describe various conditions to analyze the vulnerabilities, that can be evaluated by the OVAL interpreter. An OVAL interpreter, which is customized for each operating system, collects the detailed states of the platform, tests them against the OVAL definitions, and reports the results of the tests.

4.2.2 Linkage to Integrity Measurement

The PVA links the vulnerability information and the integrity information collected from Reference Manifests, and verifies the security state from the latest components. Vulnerability information in the XML format includes the CVE information stored in the database, and each CVE entry is associated with a security score for the CVSS.

In the IMA the files are measured by using the SHA-1 hash function. Integrity information in an IR is composed of the hash values of the files measured with the IMA. However, CVE has vulnerability information at the level of packages. In order to support file-based vulnerability detection, we extract the list of files in each package, and associate the vulnerability information of each package to each file.

Sometimes, a vulnerable version of a package and its non-vulnerable successor may include the same file with the identical hash value. In such a case, since the file did not contribute to fixing the vulnerability, the system will not associate the vulnerability with it; i.e., the vulnerability is associated with the file only when it is modified.

The report, which describes the presence of vulnerabilities and the degrees of security risks, is generated by verifying the integrity of the files and evaluating them against the vulnerability information.

Table 1 Example of security properties.

Properties	Values	Comments
Transitive Trust CRTM Type	SRTM DRTM	Platform w/ TPM v1.1b or v1.2 Platform w/ TPM v1.2 and CPU, chipset to support DRTM both BIOS and VMM must be valid
Range	both BIOS Bootloader VMM OS(IMA)	
Integrity	Black list* White list	The range of Transitive Trust is depend on the runtime implementation. Check marked components only Does not allow unknown components
Vulnerability Validation type	Deny all Severity	Do not allow any vulnerable components e.g. decided by CVSS score
Identity Credential	Issuer	AIK credential should be issued by PrivacyCA, but other TTP could be accepted by policy issued by vendor or PVA
Manifest	Issuer	

4.3 Validation Policy

The validation policy is an open issue, which is not yet standardized by TCG. In FSM based validation scheme as shown in Fig. 4, the FSM generates the security properties of the target platform, and then the policy decision point (PDP) checks the properties against a given policy. Table 1 shows an example of a set of security properties that is described in the policy, and supported by the current server-side PTS.

However, the service needs to specify the policy to express its security requirements. It is a future agenda to manage the validation policy corresponding to the policy required by the service.

5. Experimental Results

We used a commercially available PC with a TPM chip and KNOPPIX 5.1.1 for Trusted Computing Geeks [27] as the target platform. All of current Linux distributions did not support transitive trust chain from CRTM through userland components level. This modified KNOPPIX supports the trusted boot, including the transitive trust chain which covers the CRTM, BIOS, bootloader, operating system, and the user land executables. The Integrity of the user land executables and libraries are measured with the Linux-IMA. Thus, the IR generated by the client-side PTS contains all of the components that are running or ran on the target platform. Since this KNOPPIX is a CD bootable operating system with unchanging components, it simplifies the integrity management process.

Two Reference Manifests, a Platform and Runtime RM, are required to validate the target platform as shown in Fig. 5. The Platform RM covers the BIOS integrity information of PCs. Since currently no PC vendor provides such RMs, these RMs are generated by the PTS from the IML

Table 2 Experiment configuration.

Server PC (as PVA)	NEC Express5800/i110Ra-1h
CPU	Pentium M, 2 GHz
Memory	1 GB
Server OS	Red Hat Enterprise Linux4 ES
Database	PostgreSQL 7.4.1
JRE	Sun JRE 6 update 3
Web Application	Apache Tomcat 5.5.25
Web Server	Apache 2.0.52
PTS	Open Platform Trust Services 0.1.0
Client PC (as Target)	Lenovo Thinkpad T60
CPU	Intel Core2Duo 2.0 GHz
Memory	1.5 GB
Client OS	KNOPPIX 5.1.1 TC Geeks
Bootloader	Grub-IMA 1.1.0.0
Kernel	kernel 2.6.19 + Linux-IMA
TSS	Trousers 0.2.9.1
PTS	Open Platform Trust Services 0.1.1 with GCJ 4.1

collected from the running platform. The Runtime RM covers the integrity information of software components which become a part of the TCB (e.g. the Bootloader and OS Kernel). The PVA collects and collates the Platform and Runtime RMs for the evaluation.

The integrity information of the user land components is stored in the IIDB. We collected the whitelist information for the user-land components from the CD media and store it directly into IIDB, in order to replace the RM creation process and set up the IIDB for the target OS. For this KNOPPIX, the IIDB has integrity information for 84,382 files in 1,415 packages. The VULDB stores 6,536 entries of CVE data as XML published by NVD [18] from 2002 to 2007, and it also includes the corresponding CVSS scores. To validate the KNOPPIX, we obtained the Debian Security Advisory [25] from DSA-001 to DSA-1406 (published Nov 2007) with the CVE numbers. After updating the VULDB, the IIDB was synchronized with VULDB to have the latest vulnerability information.

When a user load the KNOPPIX and tries to validate it by using the PVA, The IML is verified against the RMs and the latest states are also verified according to the data in these databases. For example, if the hash value “e8df910d0bd25ebe7a0b...” is in the IML, we will know that the hash value is for the file “/usr/lib/iceweasel/firefox-bin” and a part of the “iceweasel-2.0.0.1+dfsg-1” package. This package has several vulnerabilities according to DSA-1308, such as CVE-2007-2871, 2870, 2869, 2868, 2867, and 1362. The CVSS has a range of scores from 4.3 to 9.3 in the relevant CVE entries.

After update the iceweasel to the latest version, we measure the performance. It takes around 10 seconds to validate the target platform by using the PVA via internet. And the server side validation takes around one second for about 600 measurement events. Table 2 shows the configuration used in this experiment.

6. Conclusion and Future Work

Security incidents are reported frequently, such as informa-

tion leakages of sensitive personal data, which are mainly caused by computer viruses and inappropriate management of the PCs. Maintaining and verifying the integrity of the platform configurations in a more scalable and systematic manner is essential to countermeasures against such security incidents. In this paper, we used trusted computing technology standardized in the Trusted Computing Group, where the trust model is hardware-rooted and assumes the existence of a specific security chip TPM embedded in the computing platform. We designed the Integrity Management Infrastructure based on the PTS and implemented an integrity validation mechanism based on the proposed platform behavior model that can validate the trusted-bootstrap and subsequent measurement processes of the operating systems and applications. We successfully measured the integrity of all of the binary components of KNOPPIX 5.1.1 Trusted Computing Geeks. As far as we know, this is the first successful measurement of a widely used operating system that vouches for the correct configuration of the system. We experimented with the performance of the integrity measurement and found it acceptable.

There are many limitations to apply TCG's integrity measurement and validation technology against the commercial-off-the-shelf runtime. For example, most of the applications change the behavior depending on the configuration. Therefore, in order to attest the complete platform behavior, it is necessary to measure not only the integrity of the code but also the configuration.

The digest based integrity measurement is not suitable for measuring configuration because the digest cannot capture the semantics of the configuration, which can be easily changed by the user. Thus, we have to integrate this technology with other security tools which enable semantical evaluation of software configuration. They must be included in the transitive trust chain of the platform and validated by the PVA. This becomes the delegation point of the hash based integrity protection and other semantically rich security functions.

We have tangled components dependency and intricate configurations. Future work will include broader support of other operating systems, performance optimization of the runtime validation, and incorporation of the security standard into our infrastructure.

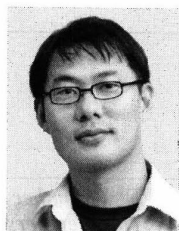
Acknowledgments

We gratefully acknowledge the support and the fruitful discussions with Toshiki Yagi, Nguyen Anh Quynh and Kuniyasu Suzuki at the National Institute of Advanced Industrial Science and Technology (AIST) about KNOPPIX. This study was sponsored by the Ministry of Economy, Trade and Industry, Japan (METI) under contract for the New-Generation Information Security R&D Program.

References

- [1] "CSI/FBI computer crime and security survey 2006."
- [2] Trusted Computing Group, <https://www.trustedcomputinggroup.org/>
- [3] W. Arbaugh, D. Farber, and J. Smith, "A secure and reliable bootstrap architecture," Proc. IEEE Symposium on Security and Privacy, pp.65-71, May 1997.
- [4] R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn, "Attestation-based policy enforcement for remote access," Proc. 11th ACM Conference on Computer and Communications Security (ACMCCS 2004), 2004.
- [5] Trusted Computing Group, "TCG trusted network connect - TNC architecture for interoperability."
- [6] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A virtual machine-based platform for trusted computing," 19th ACM Symposium on Operating Systems Principles, 2003.
- [7] Y. Watanabe, S. Yoshihama, T. Mishina, M. Kudo, and H. Maruyama, "Bridging the gap between inter-communication boundary and inside trusted components," 11th European Symposium on Research in Computer Security (ESORICS 2006), LNCS, Springer, 2006.
- [8] J.L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn, and R. Caceres, "Trusted virtual domains: Toward secure distributed services," Workshop on Hot Topics in System Dependability, 2005.
- [9] V. Haldar, D. Chandra, and M. Franz, "Semantic remote attestation — A virtual machine directed approach to trusted computing," 3rd Virtual Machine Research and Technology Symposium, May 2004.
- [10] A. Sadeghi and C. Stübke, "Property-based attestation for computing platforms: Caring about properties, not mechanisms," 2004 Workshop on New Security Paradigms (NSPW 2004), pp.67-77, 2004.
- [11] M.E. Van Herreweghen, J. Poritz, and M. Waidner, "Property attestation - Scalable and privacy-friendly assessment of peer computers," Technical Report RZ3548, IBM Research Report, 2004.
- [12] S. Yoshihama, T. Ebringer, M. Nakamura, S. Munetoh, T. Mishina, and H. Maruyama, "WS-attestation: Enabling trusted computing on Web services," in Test and Analysis of Web Services, ed. L. Baresi and E. Di Nitto, pp.441-469. Springer Berlin Heidelberg, Sept. 2007.
- [13] Trusted Computing Group, "TPM main, version 1.2."
- [14] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and implementation of a tcb-based integrity measurement architecture," 13th USENIX Security Symposium, pp.223-238, Aug. 2004.
- [15] Trusted Computing Group, "TCG infrastructure working group architecture Part II - Integrity management, Ver. 1.0, Rev. 1.0," Nov. 2006.
- [16] The MITRE Corporation, "Common vulnerabilities and exposures," <http://cve.mitre.org/>
- [17] "Open vulnerability and assessment language," <http://oval.mitre.org/>
- [18] "National vulnerability database," <http://nvd.nist.gov/>
- [19] Trusted Computing Group, "Trusted computing group, platform trust services (PTS), Ver. 1.0," Sept. 2006.
- [20] "xen.org," <http://xen.org/>
- [21] "Intel vPro technology: An active defense against security threats," <http://www.intel.com/technology/magazine/computing/vPro-security-0906.pdf>, Sept. 2006.
- [22] "Open platform trust services," <https://sourceforge.jp/projects/openpts>
- [23] Trusted Computing Group, "TCG PC client specific implementation specification for conventional BIOS, Ver 1.2," July 2005.
- [24] "Red hat network," <https://rhn.redhat.com/>
- [25] "Debian security," <http://www.debian.org/security/>
- [26] "Common vulnerability scoring system support," <http://nvd.nist.gov/cvss.cfm>
- [27] N.I. of Advanced Industrial Science and Technology, "Knoppix5.1.1 for trusted computing geeks," <http://unit.aist.go.jp/itri/knoppix/>, 2007.

[1] "CSI/FBI computer crime and security survey 2006."



Seiji Munetoh received his B.E. and M.E. degrees from the University of Tohoku, Sendai, Japan in 1990 and 1992. He joined Tokyo Research Laboratory of IBM, Japan in 1992. His research interests include VLSI design, security engineering and trusted computing technology.



Megumi Nakamura received her B.E. and M.E. degrees from Keio University in Japan at 2002 and 2004 respectively. She joined Tokyo Research Laboratory of IBM, Japan in 2004. Her current research interest includes network security, privacy protection, and trusted computing technology.



Sachiko Yoshihama is a researcher at the Security and Privacy group in Tokyo Research Laboratory of IBM, Japan. She received her M.E. in information security. Her research interests include trusted computing, information flow control, and Web security.



Michiharu Kudo is the manager of the Security and Privacy group at Tokyo Research Laboratory of IBM, Japan. He received his M.Sc. in Mechanical Engineering and his Ph.D. in Information Security with a focus on provisional access control systems. Currently he is leading several research projects on trustworthy computing platforms based on trusted computing technology and compliance-oriented security infrastructures. His research interests include access control, privacy protection, and security policies. He was one of the founders of the XACML Technical Committee in the OASIS standardization organization.