February 14, 2005

RT0598
Security 8 pages

Research Report

WS-Attestation: Efficient and Fine-Grained Remote Attestation on Web Services

Sachiko Yoshihama, Tim Ebringer, Megumi Nakamura, Seiji Munetoh, Hiroshi Maruyama

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

WS-Attestation: Efficient and Fine-Grained Remote Attestation on Web Services

Sachiko Yoshihama[†], Tim Ebringer[‡], Megumi Nakamura[†], Seiji Munetoh[†], Hiroshi Maruyama[†]
[†]IBM Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa, Japan
[‡]Department of Computer Science and Software Engineering, The University of Melbourne
[†]{sachikoy, nakamegu, munetoh, maruyama}@jp.ibm.com, [‡]tde@cs.mu.oz.au

Abstract.
This paper proposes WS-Attestation, attestation architecture on Web Services framework. We aim at providing software oriented, dynamic and fine-grained attestation mechanism that leverages TCG technologies to increase trust and confidence in integrity reporting. In addition, the architecture allows efficient binding of attestation with application context, privacy protection, as well as infrastructural support for attestation validation.

1. Introduction

Remote attestation is one of the key functionalities of trustworthy computing which allows a remote challenger to verify not only the identity of the other party but also its behavior. The trusted computing allows establishing a trust relationship among potentially distrusted distributed parties, thus enables new types of secure applications.

The Trusted Computing Group (TCG) defines a set of secure computing subsystems. The center of the TCG architecture is called the Trusted Platform Module (TPM) which is a tamper-resistant hardware module. In addition to serving as a cryptographic co-processor and a protected storage for secrets and keys, the TPM is used to measure and report platform integrity in a manner that cannot be compromised by either platform owners or the software running on it. TCG defines the trusted bootstrap process [1] [14] that comprises an iterative process of measurement, loading, and execution of software components. When the system is powered-on, the immutable initial bootstrap code measures next component and stores the measurement in the TPM before transferring control to the next component. In subsequent steps, each component recursively measures next component and records the measurements in the TPM, until the operating system is loaded. Each measurement is taken as a SHA1 secure hash value of binary image of the component, and stored into Platform Configuration Registers (PCRs). PCRs are special purpose registers within the TPM which record integrity measurements, and are protected from an arbitrary modification. A PCR supports only the *extend* operation to update its value[1][2].

Attestation is the mechanism defined in the TCG specifications to report the integrity measurements stored in PCRs. In the attestation process, TPM signs over the PCR values and the external 160-bit data (such as a nonce from a challenger) using an RSA private key, whose confidentiality is protected by TPM. The attestation is an atomic, protected operation on the TPM and the attestation signature cannot be forged by malicious software. Therefore, if the TPM is properly designed and implemented to adhere to the TCG specifications, and the platform, including the initial bootstrap code, is properly integrated with TPM, a remote verifier can have confidence in the integrity measurement reported by TPM.

This paper proposes WS-Attestation, attestation architecture on Web Services framework. It provides a software oriented, dynamic and fine-grained attestation mechanism which leverages TCG technologies to increase trust and confidence in integrity reporting. In addition, the architecture allows efficient binding of attestation with application context, as well as infrastructural support for attestation validation.

The following sections are structured as follows: Section 2 discusses design principles. Section 3 discusses architecture of attestation support for Web Services

[1] When recording a measurement value *v* into a PCR, the value is extended into the PCR, which results a SHA1 hash over concatenation of the current PCR value and the value ***v***; i.e., the new value of PCR at step *i* is `PCR(i) = SHA1(PCR(i-1) || v)`. The initial PCR value after power-on is `PCR(0) = 0`.

[2] TPM Specification v1.2 supports a new operation to reset TPM to 0. This is presumably intended to be used in Microsoft's NGSCB initiative so that a virtualized operating system can leverage TCG without a hard reset.

framework. Section 4 discusses profile of Web Services protocols for attestation. Section 5 discusses prototype implementation. Section 6 discusses related work. Section 7 concludes this paper.

2. Design Principles

This section discusses principles that are taken into account in the design of the attestation support in Web Services.

2.1. Fine granular, dynamic, verifiable, and efficient attestation

Although TCG provides a hardware-based root of trust, the platform integrity measurement and reporting it conveys little information compared with the complex state of a running system. In WS-Attestation, we aim at complementing TCG attestation with fine granularity, dynamicity, and verifiability.

Fine granularity. Trusted bootstrap, as defined in TCG, is designed to measure binary images of executables and components (e.g., BIOS configurations) during the bootstrap sequence. However, today's computing systems are complicated and include properties that cannot be meaningfully measured from their binary image. For example, behavior of Linux systems can significantly differ because of parameters specified in configuration files, even if they run on the identical OS kernel and the executable image is the same. It is not practical to measure configuration files with SHA1 hash values; as most of the Linux configuration files are text based, the system administrator can easily break the integrity of a configuration files by adding a white-space or a blank line, even though the semantics of the configuration file is not changed. Therefore it is important that attestation can provide not only binary measurements but also semantic information, e.g., platform configuration retrieved by a software-based attestation agent.

Dynamicity. Trusted bootstrap measures integrity of executable components up to the operating system. However, various executables and data loaded on the operating system and on the application layer affect behavior of a running system [3]. It is important that the WS-Attestation can support rich semantic attestation information while leveraging root-of-trust defined in TCG.

Verifiability. TPM stores measurement of components in PCRs in the form of composite hash values. Each composite hash value represents a list of components that are measured and 'extended' into a PCR. It is assumed that one PCR is used to measure quite a few components; e.g., TCG defines minimum 16 PCRs for PC platforms, and 8 of them are reserved for measuring BIOS, while the other 8 are used for measuring the OS and the application layer. As the number of components measured by a PCR becomes bigger, and as the number of possible revisions of each component becomes bigger, the number of permutations that constitute a PCR value becomes factorial; It becomes very difficult for a verifier to validate the platform integrity from a PCR value.

Efficiency. As information conveyed and validated in an attestation becomes more detailed, the attestation process can become overly expensive. On the other hand, we cannot simply separate attestation from the application context, because an entity sending an application message may not be in the same state as what was attested, thus may not be trusted anymore. It is important to increase efficiency while maintaining a cryptographic binding between attestation and application context.

2.2. Attestation Supporting Infrastructure

As a large number of vulnerabilities are found every day [5], software vendors release security patches quite frequently. A typical security patch consists of multiple files that replace vulnerable components on the system. Each patch may fix one or more vulnerabilities. Thus it becomes increasingly difficult to make educated decisions as to whether vulnerability is present in a particular file. A well organized infrastructural support is therefore essential to enable validation measurement of each component on the system.

Finally, each entity requesting attestation may not be capable of validating attestation information. We assume presence of trusted third party validation services that validate attestation on behalf of requesters. We aim at defining communication models between the attestation requester, responder, and the validation service.

2.3. Privacy Protection.

There are two types of privacy need to be considered in attestation: *identity* and *integrity* of the platform being attested.

Identity Privacy. It has been one of the key objectives of TCG attestation to protect privacy of platform identity while establishing trust. TCG defines two mechanisms for identity privacy: the Privacy-CA and Dynamic Anonymous Attestation (DAA). Since current TCG specifications already address identity privacy issues, we do not focus on the identity privacy in this paper.

Integrity Privacy. The most unique aspect of attestation is that it proves not only the identity of the platform but also the integrity and state of the platform. Although it is useful information for a legitimate verifier to judge trustworthiness of a platform, it might also become a source of vulnerability if distrusted parties can perform attestation. For example, by investigating OS version and applied security patches, an attacker can quickly deduce

the most effective attack techniques. Therefore, it is important, especially in cross-organizational transactions, that a platform can prove its trustworthiness to anonymous challengers without disclosing its configuration details. This is addressed in section 3.5.

3. WS-Attestation Architecture

Figure 1 shows architecture of attestation support on Web Services. The *attested platform* is a platform that is being attested. The *attestation requester* initiates attestation request, which may or may not be able to validate attestation response by itself. The validation service is a trusted third party authority that validates (or sometimes performs) attestation on behalf of the requester. The validation service refers to the integrity database for validating integrity of each component measurement. The Privacy CA or the DAA issuer is responsible for certification of AIKs generated on attested platforms.

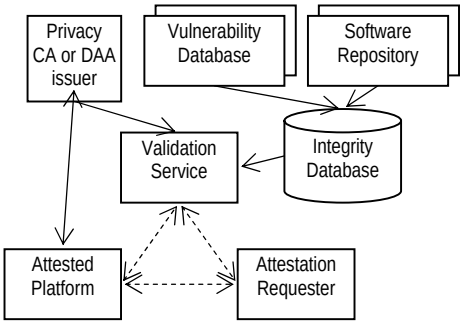

Figure 1 WS-Attestation Architecture

3.1. Attested Platform

The platform being attested implements various forms of integrity measurements and is capable of responding to an attestation request. It is also assumed that the attested platform implements appropriate security mechanisms and policies that is to be required by the attestation requester, and presence of such implementation can be measured and reported in the attestation process.

Integrity measurements consist of the following mechanisms.

TCG Trusted Boot. TCG trusted boot starts measurement from the hardware-based root of trust, and measures all components up to the OS.

Run-time measurement at OS. While the system is running, various behavior, such as module loading or application execution, are monitored and measured by the operating system and recorded into PCRs. Integrity Measurement Architecture [3] realizes such measurement on the Linux kernel.

Run-time measurement at Middleware. Various forms of middleware constitute today's computing systems. However, it is not practical to extend OS to measure integrity of data that are used by middleware, because that requires rebuilding OS each time when needing to support a new type of middleware or data. We think that it is desirable that each middleware layer measures data that is loaded or used by itself. An example of the measurement at middleware is a Java™ Virtual Machine (JVM) that measures integrity of Java class files when each class is loaded.

Care needs to be taken, though, that a chain of trust needs to be maintained from the root-of-trust to the component being measured. That is, 1) the integrity of the base code up to OS is measured in the trusted bootstrap sequence, 2) the integrity of a middleware is measured by OS, 3) and finally, the integrity of a file being loaded by the middleware is measured by the middleware. The record of measurements (stored in TPM) must prove that each component is measured by a component that is already measured, and the measurement record is not forgeable.

Measurement by Agents. System properties that are not measured by the binary measurement may be measured and reported by an agent. An example of such an agent is a local daemon that reads system configuration files, and composes a structured message that describes the properties of the configuration (e.g., network setting, minimum password length, etc.). Similar to the middleware level measurement, the chain of trust from the root-of-trust to agents needs to be maintained.

3.2. Attestation Measurements and Credentials

There are several forms of information exchanged in attestation processes that are different in levels of confidence and granularity.

Attestation Signature. The TCG attestation signature is an RSA signature value generated by an AIK over concatenation of the target data and PCR values. Since the signature operation is an atomic operation performed by TPM, and values in PCR and use of the AIK is also protected by TPM, a TCG attestation signature proves that the signed PCR values are not compromised, and represents the state of the attested platform at the time of signing.

Platform Measurement Description (PMD). The PMD is structured data that describes the state of the platform in a fine-grained and semantic manner. A PMD would include the log of measurements that are recorded during the trusted bootstrap and run-time, to describe which components have been measured by PCRs. Such a log

allows the verifier to validate integrity of every component running on the system. The verifier can also verify that hash of all components in the log matches the PCR values in the attestation signature. Since the PCR values in the attestation signature are not forged, as long as TPM is genuine and not in direct contact with an attacker who performs hardware-level attacks, we can use these values to verify the PMD that is generated by potentially distrusted software.

Attestation Credentials. As PMDs become richer, validating the PMD at each transaction may take too much time and becomes a bottleneck. To realize efficient attestation, we propose the notion of attestation credentials. An attestation credential has properties that are asserted by an authority, and may have expiration period. A typical attestation credential is issued by a trusted authority that asserts some properties (e.g., hasKnownVulnerability='false') about an attested platform. An attestation credential may bind a particular set of PCR values to the properties. Upon a challenge by an attestation requester, the attested platform may present the attestation credential along with the attestation signature signed over the challenge. By verifying the challenge, the PCR values and attestation credentials, the attestation requester can verify, without knowing the details of measurement description, that the attested platform's current state is represented by the PCR values in the attestation signature, and the PCR values represent the properties that are asserted in the attestation credential. The attestation credential also help protecting integrity privacy of the attested platform from potentially distrusted attestation requesters, especially by utilizing PCR obfuscation technique described in Section 3.5.

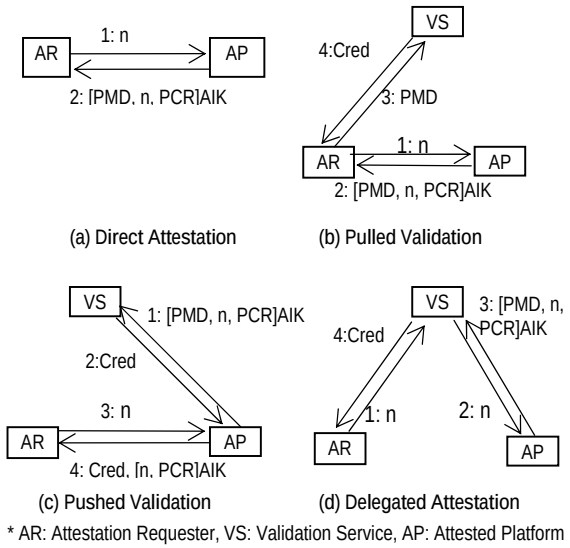

Figure 2 Attestation Models

3.3. Attestation model

An attestation requester (AR), an attested platform (AP), and a verification service (VS) play central roles in an attestation, especially in verification of integrity of the attested platform. This section discusses four attestation models each of which is built on a different trust model, and has advantages and disadvantages.

Direct Attestation. Figure 2 (a) shows the Direct Attestation Model in which an attestation requester challenges the attested platform, which then returns the measurements back to the requester. The attestation requester validates information by itself, which has the advantage of not requiring that any other party need be trusted. This model has two notable disadvantages. 1) The attestation requester has to be capable of validating the attestation response; 2) the attested platform has to disclose all of its integrity information to the requester, which violates its integrity privacy to potentially distrusted attestation requesters.

Attestation with Pulled Validation. The second model (Figure 2 (b)) is similar to the Direct Attestation, except that the attestation requester consults the validation service to validate the PMD, and does not have to be capable of validating attestation. Integrity and privacy of the attested platform is not protected in this model. An additional disadvantage is that this model may suffer from the performance bottleneck of the validation service, because for every attestation the validation service needs to be contacted.

Attestation with Pushed Validation. In the attestation with pushed validation model (Figure 2 (c)), the attested platform *pushes* the attestation to the validation service, to request an attestation credential. Upon a challenge from the attestation requester, the attested platform sends the attestation credential along with the attestation signature over the challenge, thus allowing the attestation requester to verify that the attested platform has the properties asserted in the credential. The advantages of this model are that 1) the attested platform does not have to disclose integrity information to the attestation requester; 2) the attestation requester does not have to be capable of validating attestation, 3) the performance bottleneck at the validation service is of less concern, because once an attestation credential is returned from the validation service, the attested platform can re-use the credential for subsequent transactions. Finally, the attested platform can choose which validation service to disclose its integrity information to, thus helping maintain the privacy of platform.

Delegated Attestation. In the delegated attestation model (Figure 2 (d)), the attestation requester requests a validation service to perform attestation on behalf of the requester, and then send only the validation result in the

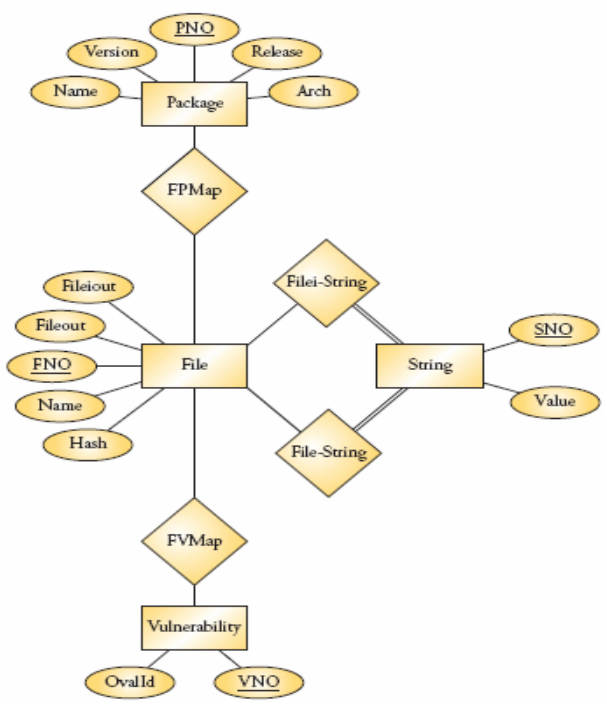

Figure 3 Integrity Database ER Diagram

form of a credential. The advantages of this model are that 1) the integrity privacy of the attested platform is protected; 2) the attestation requester does not have to be capable of validating attestation.

3.4. Attestation Supporting Infrastructure

One infrastructure for supporting attestation we constructed was the *integrity database*, which allows attestation verifiers to query integrity and vulnerability of each measured component.

Many OSes support mechanisms to distribute software components and patches in precompiled packages. For instance, RedHat's Package Manager (RPM) is the standard way of distributing and deploying components of RedHat's Linux distribution. When a different version is then distributed, the executable images in the package almost always have a new hash value. Thus, the exact version of an RPM package can often be deduced from the hash values of its executable files.

A relatively recent endeavor in platform security is the Online Vulnerability and Assessment Language [11], sponsored by MITRE and supported by various operating system vendors, including RedHat. OVAL is a language for expressing the preconditions necessary for a vulnerability to exist. Although the exact semantics differ depending on the operating system platform, the RedHat variant references particular RPM packages.

A hash database of RPM packages was built by simply unpackaging RPMs and generating hash values of all ELF executables. By parsing OVAL vulnerability descriptions and correlating these with RPM package versions, we were then able to deduce which executable hash values would indicate the presence of vulnerabilities.

We found that a verifier with a database like the one could verify the RPM-providence of all the executable images loaded. Furthermore, by cross-referencing with OVAL vulnerabilities, they could determine the presence of vulnerabilities, merely from the hash values.

3.5. Privacy Protection: PCR Obfuscation

As we discussed in Section 2.3, attestation may need to address two types of privacy issues: *identity* and *integrity*. This paper focuses on privacy of integrity information. One problem of attestation is that it provides detailed configuration information that is very useful to an attacker, since they may instantly learn which attack tools will be effective against the platform, or when a platform has changed its configuration. The solution to this problem is to extend each PCR register with a random value at random times, yet at the same time recording these random values in the log. The resulting PCR value is unpredictable; provides no information about configuration details to the attacker. However, with the log of all measurement by PCR, a legitimate verifier can still verify integrity of all other components, and that the current PCR value matches with what is derived from the log, including extra random extensions.

In the attestation models with a third-party validation service, the validation service may issue a credential to the measurement log including random extensions, and the credential asserts that some properties are true only when the attested platform has a particular set of PCR values. The attestation requester that receives the credential and the current PCR values cannot derive detailed configuration from PCR values, but it can verify that the current PCR values prove the properties asserted in the attestation credential.

Random extension of PCRs may be performed any number of times, as long as the log of extension is maintained. Especially important is that the extension is performed more frequently than release of security patches components that run on the system. If a patch that fixes vulnerability is released, by observing PCRs before and after the patch release, an attacker can infer whether the patch is applied to the platform and will be able to make an educated decision on attack tactics.

3.6. Secure Context Establishment

Several approaches are possible to bind the state of an attested platform to an application context.

First, the attestation requester and the attested platform may establish a secure communication channel before attestation. WS-Trust [6] defines a key exchange protocol to exchange a shared secret, which enables the binding between attestation and subsequent transactions by adding Hashed Message-Authentication Code (HMAC) to the messages. Care needs to be taken to protect the shared secret from being bound to the distrusted attested platforms; not only must the attestation requester discard the shared secret when the attestation fails, but the attested platform must also discard the shared secret when its state changes. Especially when the application is terminated, or the system is rebooted, the attested platform must exchange a new shared secret and start the attestation process again; it must be verified before a key exchange that the attested platform and its applications are implemented to relinquish shared secrets at termination. However, if the state of the attested platform changes without terminating the application, e.g., as a result of additional kernel module being loaded, change of such state is difficult to detect at the application layer. To prevent the use of a shared secret in a context that is not expected, the secret should expire and be renewed in an every short window of time. This has the obvious side-effect of reducing performance.

Second, on each application message, the attested platform can present the attestation credential to the requester along with the attestation signature. The PCR composite hash value included in the attestation signature proves the state of the attested platform at the time of signing, and therefore that the properties asserted in the credential are still in effect. The freshness of the attestation signature has to be verifiable; e.g., by having a signature over the timestamp and the application message body. If the attestation signature is performed on a SOAP response message, the entire application protocol should include a challenge-and-response scheme. Although performing attestation signature on each message requires extra processing power on each party, this mechanism allows verifying the latest state of the attested platform without needing to maintain shared secret keys between peers. Once an attestation credential is issued, the credential can be re-used until it expires or is revoked. An attestation credential may be valid even for multiple attested platforms as long as they have identical integrity measurements.

4. WS-Security Attestation Profile

In order for WS-Attestation to be widely adopted and interoperable, it is considered to be important that WS-Attestation matches the model and framework of the existing WS-Security standards. Therefore, rather than invent a new protocol for attestation, we leverage existing Web Services standards and define a profile for supporting attestation on top of these standards.

4.1. Attestation Signature

Attestation signature, which is generated by TPM_Quote operation of the TPM, is considered a special form of the RSA signature. In order to handle the attestation signature in WS-Security as an XML digital signature, we define a new signature method. The method is identified by the URI and specified in the `Algorithm` attribute of the `SignatureMethod` element of the XML digital signature.

In order to verify an attestation signature, the verifier needs to be informed of the TPM_QUOTE_INFO structure that is being signed. In the proposed signature method, this structure is concatenated to the signature value that is included in the SignatureValue element as TPM_QUOTE_INFO || [TPM_QUOTE_INFO]AIK (where || denotes concatenation and [x]Key denotes a signature over x with the Key).

In the XML Digital Signature which defines extensible XML schema, it is also possible to add the `TPM_QUOTE_INFO` structure as a separate XML element. However, adding this structure to the signed value has two advantages. First, the same signature method can be used in protocols other than WS-Security where messages have no or little extensibility to include an additional element of information. Second, provider-model crypto API such as Java Cryptographic Extension (JCE) [7] supports different crypto algorithms under the same generic API. Such generic API cannot be extended to add an extra parameter without losing advantage of plugability. By including the `TPM_QUOTE_INFO` in the signature value, the crypto provider can receive the necessary information for verification of an attestation signature through a generic API.

4.2. Attestation Token

To communicate integrity measurements attestation credentials on WS-Security framework, three types of security tokens are defined.

Measurements. The simplest form of an attestation token conveys binary measurements recorded in PCRs. The measurement token can be used to provide a list of PCR values. On verification, the verifier should verify that the PCR values match the composite PCR hash in the attestation signature.

Platform Measurement Description. The Platform Measurement Description (PMD) conveys finer grained and more semantic information of the state of the platform being measured. An example of a PMD includes lists of components and their SHA-1 hash values that are measured by TPM. In addition, a PMD may include properties of a platform that are not measured by TPM;

for example, operating system configurations and parameters that are read by an attestation agent on the platform.

Attestation Credentials. A credential may be issued by a trusted third party to assert some properties of an attested platform. A credential may be identity-based, integrity-based, or both. An attestation credential refers to a credential that asserts some properties of an entity that possesses particular measurements. For example, an attestation credential may assert the level of trustworthiness of an attested platform which PCR has a particular set of values. An integrity-based credential can be represented in various forms; e.g., an X.509 attribute certificate and a SAML Assertion [8] are well-standardized formats for this purpose.

4.3. Attestation via WS-Trust

Each of attestation models consists of credential exchange between the attestation requester, the attested platform and the validation service. Rather than defining a proprietary protocol for attestation, we leverage WS-Trust [6]. In a WS-Trust message, a requester may request a particular type of a security token, with an optional challenge. Upon a successful response, the responder returns the requested security token. The challenge in the request should be returned back to the requester with a responder's signature over it, thus proving that the response is fresh and is not replayed from past records. WS-Trust messages can be used in each of messaging in four attestation models. For example, in the delegated attestation model (Figure 2), four exchanged messages are structured as follows (described in an informal format).

```
1: Security Token Request from AR to VS
       TokenType(Attestation Credential)
       Challenge(n1)
2: Security Token Request from VS to AP
       Token Type(PMD, PCR), Challenge(n2)
3: Security Token Response from AP to VS
       [Token(PMD, PCR), Challenge(n2)]AIK-AP
4: Security Token Response from VS to AR
       [Token(Attestation Credential)
       Challenge(n1)]KeyVS
```

5. Prototype

Figure 4 shows the structure of the prototype system.

The integrity of the Linux OS is measured by the modified boot loader, and loadable modules and executables are measured by IMA (version?). The measurements (SHA1 hash values of files) are stored in PCRs as well as in the kernel-held measurement list.

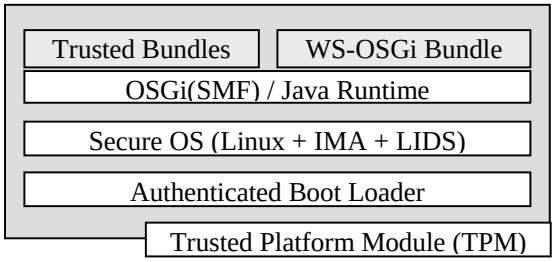

Figure 4 Prototype System

Linux Intrusion Detection System (LIDS) is used to improve the OS level security of client machine. LIDS is a kernel patch and admin tools which enhances the OS security by enforcing Mandatory Access Control (MAC) policies on operating system resources.

The prototype service is implemented in Java, and runs on the OSGi (Open Service Gateway initiative) platform, which is an open-standard framework for Java based applications and services. We extended IBM Service Management Framework (SMF), one of the OSGi implementations, to measure each bundle JAR file when it is loaded and record the measurement into PCR and the log.

We also extended the WS-OSGi, light-weight SOAP/WS-Security engine for OSGi platforms, to support the attestation signature tokens described in Section 4. The attestation signature and verification operation are implemented as a JCE crypto provider, thus allows WS-Security engine to switch between an ordinary RSA signatures the attestation signature simply by specifying the signature algorithm and the key storage as a set of options. The attestation requester, attested platform, and validation services are implemented as services on the OSGi platform and communicates each other using the WS-Trust protocol. PMDs returned by the attested platform consists of stored measurement log in the XML format, while an attestation credentials are implemented as a SAML attribute assertion signed by the validation service. The integrity database is built on DB2 and queried by the validation service by SQL over JDBC. The integrity database currently supports RPM packages only; data entries are generated from RPM package repository for RedHat Enterprise Linux 3 (REL3) and OVAL repository for this OS, thus capable of validating integrity of REL3 systems.

6. Related Work

Related work includes previous efforts to establish trust relationship between parties measuring, reporting and verifying system integrity.

AEGIS system by Arbough et al [2] provides secure bootstrapping architecture on PC system that maintains

integrity chain from the lowest trustable layer of a system. Secure bootstrap is different from trusted bootstrap in a sense that its objective is not to allow remote verification of the system integrity; in the secure bootstrapping, the system aborts bootstrap process upon integrity check failure.

Sailer et al leverages TCG in Integrity Measurement Architecture (IMA)[3], to enhance the role of the TPM not only to measure static state of a system but also dynamic state. IMA is implemented as a Linux Security Module to measure each executable, library, or kernel module upon loading and record the SHA1 hash values into TPM and the log. As mentioned earlier, we leverage TCG and IMA to build Linux based attested platforms.

More recent work of Sailer [4] utilizes the integrity measurements and attestation to protect remote access points, to enforce corporate security policies on remote clients in a seamless and scalable manner. Cisco and IBM have announced an enterprise network security solution [12] based on their current products: Cisco's Network Admission Control (NAC) protects the network infrastructure by enforcing security policy compliance on all devices seeking to access network computing resources. The integrated security solution leverages IBM Tivoli Compliance Manager (TSCM) which inspects device configurations, thus denies network access to the devices that are not compliant to the corporate security policies. The compliance checks are based on software agents (e.g., whether anti-virus software is up to date, or the OS is running the latest software patches), but NAC's extensible architecture would allow incorporating further attestation mechanisms in the future.

Terra by Garfinkel et al [13] realizes isolated trusted platforms on top of a virtual machine monitor, and allows attestation by a binary image of each virtual machine, e.g., virtual disks, virtual BIOS, PROM, and VM descriptions.

Recent efforts on mitigating drawbacks of TCG attestation include Haldar's proposal [9], which leverages language-based security and virtual machines to enable *semantic attestation*, e.g., attestation of dynamic, arbitrary, and system properties as well as behavior of the portable code. Property-based Attestation [10] by Sadeghi and Stüble proposes an attestation model with a trusted third party that translates low-level integrity information into a set of properties.

7. Conclusion

This paper presents our proposal on WS-Security support for attestation. Although attestation is a generic technique to allow remote verification of platform integrity, our proposal is based on TCG, which is the most promising and available technology at the moment of this writing. This paper shows a set of profiles that seamlessly works on existing WS-Security standards. We also take privacy protection into account, as well as provide infrastructural support for efficient, accurate, and fine granular attestation validation.

Acknowledgements

The authors wish to thank many people of IBM Corporation for comments and insights on an earlier version of this paper.

References

1. Trusted Computing Group, TPM Main Specification http://www.trustedcomputinggroup.org/
2. Arbaugh, W.A., Farber., J., Smith., J.M., A Secure and Reliable Bootstrap Architecture, in IEEE Computer Society Conference on Security and Privacy. IEEE, 1997, pp.65-71.
3. Sailer, R., Zhang, X., Jaeger, T., Van Doorn, L., Design and Implementation of a TCG-Based Integrity Measurement Architecture, IBM Research Report RC23064, January 16, 2004.
4. Sailer, R. et al., Attestation-based Policy Enforcement for Remote Access, IBM Research Report RC23205, May 4, 2004.
5. http://www.cert.org/stats/cert_stats.html
6. Web Services Trust Language (WS-Trust), http://www-106.ibm.com/developerworks/library/specification/ws-trust/
7. Java Cryptographic Extension (JCE), http://java.sun.com/products/jce/
8. Assertion and Protocol for the OASIS Security Assertion Markup Language (SAML), http://www.oasis-open.org/
9. Haldar, V., Chandra, D., Franz, M., Semantic Remote Attestation – A Virtual Machine directed approach to Trusted Computing, the 3rd Virtual Machine Research and Technology Symposium, May 2004.
10. Sadeghi, A., Stuble, C., Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms, New Security Paradigm Workshop (NSPW), 2004
11. Open Vulnerability and Assessment Language, http://oval.mitre.org/
12. IBM and Cisco: White Paper, October 2004, http://www-3.ibm.com/security/cisco/docs/wp-ibm-cisco-iisfcn.pdf
13. Garfinkel, T., et al., Terra: A Virtual Machine-Based Platform for Trusted Computing, the 19th ACM Symposium on Operating Systems Principles, 2003.
14. TCG Specification Architecture Overview, Revision 1.2 28 April 2004.