Trusted Linux Client and Trust Aware Web Sewer

Developed By:

Tamleek Ali Tanveer

(15-FAS/PHDCS/04)

Supervised by

Prof. Dr. Khalid Rashid

Department of Computer Science

Faculty of Basic Applied Science

International Islamic University Islamabad

2008

A thesis submitted to the

Department of Computer Science

International Islamic University Islamabad
As a partial fulfillment of requirements for the award of
The degree of

MS in Computer Science

International Islamic University, Islamabad

Dated: 7.8.2008

Final Approval

It is certified that we have examined the thesis titled "Trusted Linux Client and Trust-aware Web Server" submitted by Tamleek Ali Tanveer, Registration No. 15-FAS/PhDCS/04, found as per standard. In our judgment, this research project is sufficient to warrant it is acccptancc by the International Islamic University, Islamabad for the award of MS Degree in Computer Science.

Committee

External Examiner
Dr. Muhammad Masoom Alam
Assistant Professor,
Institute of Management Sciences, Peshawar.

[illegible]

Internal Examiner
Prof. Dr. Muhammad Sher
Chairman,
Department of Computer Science
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad

[illegible]

Supervisor
Prof. Dr. Khalid Rashid
Head, Department of Computer Science,
COMSATS Institute of Information Technology,
Islamabad.

[illegible]

Dedicated to ONE

Who has all the names, and who does not need any name

Declaration

I hereby declare that this work, neither as a whole nor as a part has been copied out from any source. It is further declared that we have proposed and implemented an integrity based access control mechanism in the web servers on the basis of my personal efforts and under the sincere guidance of my supervisor Prof. Dr. **Khalid Rashid.** If any part of this project is proved to be copied out from any source or found to be reproduction of some other person, I shall stand by the consequences. No portion of the work presented in this dissertation has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Tamleek Ali Tanveer

(15-FAS/PHDCS/04)

Acknowledgment

All praise to Almighty Allah who has guided me to undertaking work on a Trusted Linux client and Trust aware web server and has helped me through at each step when I felt there was no hope of pulling through.

I am very obliged to my supervisor Prof. Dr. Khalid Rashid to provide me technical help for each and every problem I faced during my research work and provided me each and every information about research methodology and always guided me what to do next.

I would also like to thank our colleagues and faculty members of the University especially to Dr. Muhammad Sher and Mr. Qaisar Javed for their help and support, with special thanks to Mr. Noman for helping me in coding and implementation of the prototype.

Finally I owe a lot to my beloved parents (late) and my family for their love, guidance and support.

Project in Brief

Project Title:	**Trusted Linux Client and Trust Aware Web Server**
Undertaken By:	**Tamleek Ali Tanveer**
Supervised By:	**Prof. Dr. Khalid Rashid**
Start Date:	**September 2007**
Completion Date:	**May 2008**
Tools and Technologies:	**Trusted Java, Linux, Eclipse, JDK 1.6, MySQL, JDBC.**
Documentation Tools:	**MS word, MS Excel**
Operating System:	**Linux Fedora Core 6**
System Used:	**Pentium 4, 1.73 GHz**

Abbreviations

AIK	Attestation Identity Key
BIND	Radio Frequency
BIOS	Basic Input Output System
CPU	Central Processing Unit
EK	Endorsement Key
GRUB	GRand Unified Bootloader
HTTP	Hyper Text Transfer Protocol
IMA	Integrity Measurement Architecture
JTSS	Java Trusted Software Stack
MBA	Model-based Behavior Attestation
NGSCB	Next Generation Secure Computing Base
PCR	Platform Configuration Register
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
RSA	Rivest Shamir Adleman (Public key cipher algorithm)
SAML	Security Assertion Markup Language
SHA-1	Secure Hash Algorithm 1
SML	Stored Measurement Log
SOAP	Simple Object Access Protocol
SP	Service Provider
SR	Service Requester
SSL	Secure Socket Layer
TC	Trusted Computing
TCG	Trusted Computing Group
TLC	Trusted Linux Client
TNC	Trusted Network Connect
TPM	Trusted Platform Module
TSS	Trusted Software Stack
VPN	Virtual Private Network
VS	Validation Service
WS	Web Service
WSDL	Web Service Description Language
XML	Extensible Markup Language

Table of Contents

List of Figures

1. Introduction

1. Introduction

Recently, attackers have shifted their focus from sniffing networks and targeting servers to focusing on client machines. Survey shows that more than half of the desktop computers are infected with viruses [1]. In the last six months of 2007, Symantec detected 499,811 new malicious code threats (Fig 1.1). This is a 136 percent increase over the previous period, when 212,101 new threats were detected, and a 571 percent increase over the last half of 2006. In total, there were 711,912 new threats detected in 2007, compared to 125,243 threats in 2006, an increase of 468 percent. This brings the overall number of malicious code threats identified by Symantec to 1,122,311, as of the end of 2007. This means that almost *two* thirds of all malicious code threats currently detected were created during 2007. Safford [2] points out that:

> ***"In the 80s, hackers largely attacked the network, passively sniffing passwords, and acrively hijacking network sessions. As applications increasingly encrypted data going across the network, hackers then turned their attention largely to attacking servers directly, mainly through misconfigured or buggy services, like web servers. As companies have responded with firewalls, intrusion detection, and security auditing tools to protect their servers, hackers have increasingly turned to hacking clients':***

Trojans are also frequently used to steal information that an attacker can sell or profit from in other ways. For example, the Gampass Trojan [1] can be used to steal a user's online gaming account information, which can then be sold to other garners. Similarly Silentbanker Trojan [3] can be used to steal users' online banking credentials and divert legitimate transactions. This Trojan includes sophisticated mechanisms to steal funds from a user's online banking.
Report from symantec shows that between July 1 and December 31, 2007, they observed an average of 61,940 active bot-infected computers per day (Fig 1.2), a 17

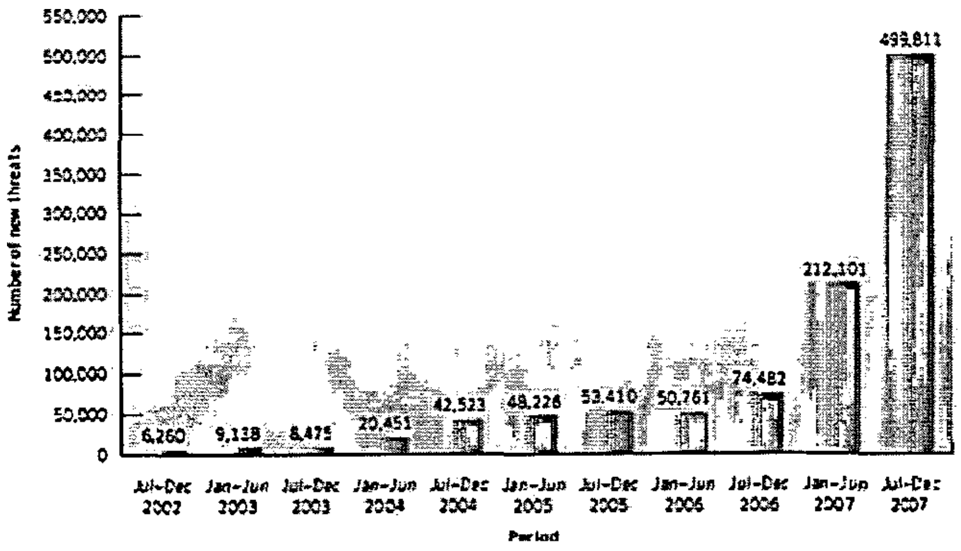


Figure 1.1:* *New malicious code ***threats** (source[1])*

percent increase from the previous reporting period. An active bot-infected computer is one that carries out an average of at least one attack per day. Symantec also observed 5,060,187 distinct bot-infected computers during this period, a one percent increase from the first six months of 2007. A distinct bot-infected computer is a distinct computer that was active at least once during the period [1]. It means that the number of Trojans is increasing day by day and threat to confidential data remains on the client side. We cannot deny access for some confidential data from any client platform because workers or even executive staff working at remote sites needs access to company's confidential data while they are on travel.

Web browser is a critical and ubiquitous application that has become an increasingly popular subject for vulnerability researchers over the past few years. Traditionally, the focus of security researchers has been on the perimeter sewers' firewalls, and other assets with external exposure. However, client-side vulnerabilities are now becoming a focus for research and attacks, alike. As part of this shift toward client- side issues, vulnerabilities in Web browsers have become increasingly prominent, which in turn poses a threat to end-users and to the enterprises equally.

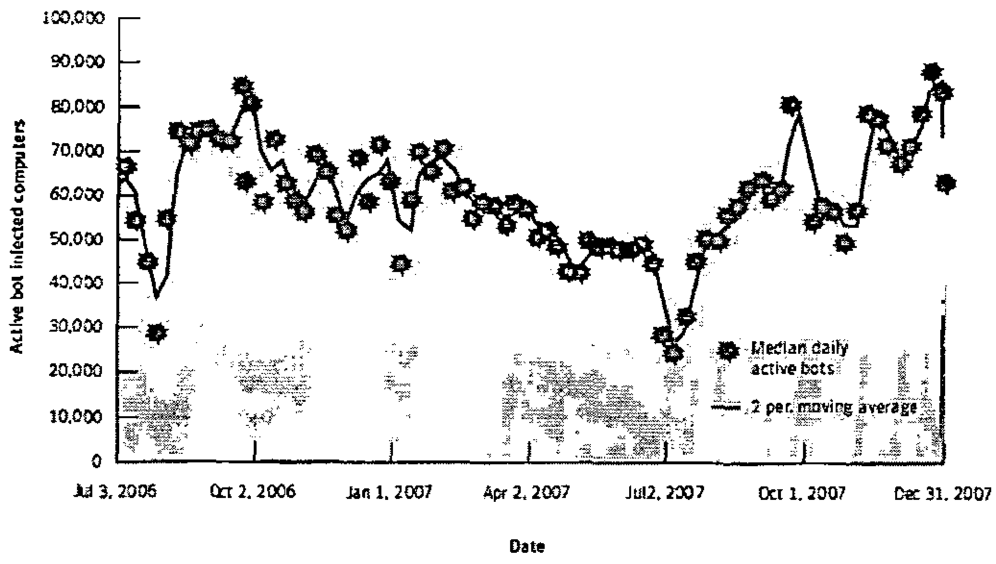


Figure 1.2; Active bot-infected computers by day (source [1])

Web browser vulnerabilities are a serious security concern due to their role in online fraud and the propagation of spyware and adware. They are particularly prone to security concerns because they come in contact with more potentially untrusted or hostile content than most other applications. Similarly in the last six months of 2007, threats to confidential information made up 68 percent of the volume of the top 50 malicious code samples causing potential infections (Fig 1.3). This is an increase over the 65 percent reported in the first half of 2007 and the 53 percent From the same period in 2006.

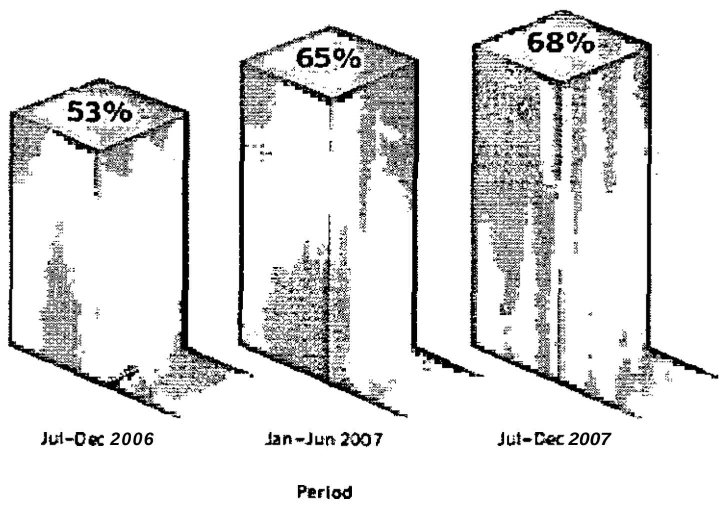


Figure 1.3: Threats to confidential information by volume (source [I])

We cannot fix the current client software. There are many reasons, including complexity, compatibility, and compromise. First of all, modem systems are incredibly complex. A typical UNIX or Windows system, including standard applications, represents something around 100 million lines of source code. Large applications often also have hundreds of millions lines of code. All together, there are billions of lines of code in use today.

Several recent studies have shown that typical product-level software has roughly one security-related bug per thousand lines of source code across its lifetime. Thus, a typical system will potentially have a hundred thousand security bugs. Survey shows that more than six thousand bugs are found per year, and that the rate of finding them has increased dramatically.

Second, compatibility requirements in the client space make a complete break from the commitment to the current system architecture uniikely. Even if we could build secure software systems, the amount of effort needed to replace the billions of lines of code in and for existing operating systems is simply prohibitive.

Third, without hardware support, it is likely impossible to detect the presence of malicious code in a system. This question is still an active area of research; but so far, all attempts to detect malicious changes in software (compromise) without hardware support have ultimately been circumvented. In contrast, with a little bit of hardware support, it is quite easy to detect compromise.

Existing web sewers' access control and authentication mechanisms does not provide any information about the status of the client machine. It does not know whether a malicious software such as Trojan, virus or bots running on the client platform and can ultimately take the confidential data from the client. Trusted Computing Group [4] is a consortium that addresses these issues. They provide mechanisms to protect systems fiom these malicious software and if any is running on the machine a remote system can detect this misbehavior. In the following section we discuss the concepts and technologies that are related to these issues.

1.1 Background

To understand these concepts we provide description of the related concepts and technologies such as Trusted Computing, Remote Attestation, Integrity Measurement Architecture and some existing web access control mechanisms briefly.

1.1.1 Trusted Computing

The term "Trusted Computing" refers to a technology in which PCs, consumer electronic devices, PDA's and other mobile phones are equipped with a special hardware chip to use cryptographic mechanisms to certify the integrity of the (application/system) software running on the device and to protect I/O and storage of data inside the device. This practice is designed to effectively fight against malicious code, viruses and privacy issues. The reason is that current practices for fighting against the malicious code and other threats are purely at the software level. The software solutions are always prone to be compromised one way or the other. It has been learned from the past experiences that a trusted and tamper proof security basis cannot be achieved using software based solutions alone [2].

In order to shape these efforts major hardware vendors have formed a consortium known as Trusted Computing Group [4]. The goal is to create Trusted PCs, PDA's and mobile phones in order to make all e-applications (e.g. e-commerce, e-health and e-govt. etc) more trust worthy. According to the specifications for TC-enabled systems, they implement the following four technical functionalities:

- Protected I/O : All information sent via I/O devices such as monitors, sound cards should be encrypted in a way that only intended applications can capture this information
- Curtaining Memory: Data stored in the memory should be strictly isolated from other applications by means of encryption.
- Sealed Storage: Permanent storage devices such as hard-drive should be in a sealed (encrypted) form so that only the originating application or device can read it. The objective is that if the data is moved from the sealed storage maliciously to another device, the data is not readable any more.
- Remote Attestation: The most important functionality of a rusted platform is to remotely certify to third parties in enciphered form, which software is running, whether malicious code has modified the corresponding software,

status of the hardware components etc. This feature enables the service providers to deploy their services across geographical boundaries.

Experts in information security conclude that some security problems cannot be solved by software alone [2], and even conventional secure operating systems depend on hardware features to enforce separation of user and supervisor modes. Trusted computing provides confidence in computing platform's behavior to the owner of a company PC, and also provides confidence in the platfonn's behavior to the individual user of that PC. Similarly, TC provides confidence in a platform's behavior to local user and provides that confidence to a remote entity across a network through remote attestation.

When a system is booting, first the BIOS is in control. That system establishes enough subsystems to do basic input and output and initialize cards. Then it transfers control to the BIOS of various installed cards and receives control back, control is passed to the boot loader, which in turn passes control to the OS kernel. The OS loads various device drivers and services and then may start up a program, or be asked to start program by the end user. By the time the end user finally direct the OS to do something; control over the machine has transferred hehveen a number of different items. In this situation, how can a user know that the system has not been hacked any number of times to allow a cracker access to everything he is doing or having on his disk.

I.1.2 TPM

The key component in the trusted computing architecture is the Trusted Platform Module (TPM) chip which provides secure storage for data and cryptographic keys. Each TPM is uniquely identified by its Endorsement Key (EK) that is burned in to it by its manufacturer. According to [5], the endorsement key is a 2,048-bit RSA public and private key pair, which is created randomly on the chip at manufacture time and cannot be changed. The private key never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip". An Attestation Identity Key (AIK) – generated by the TPM and signed by a trusted third party, uniquely identifies the particular users of a target platform.

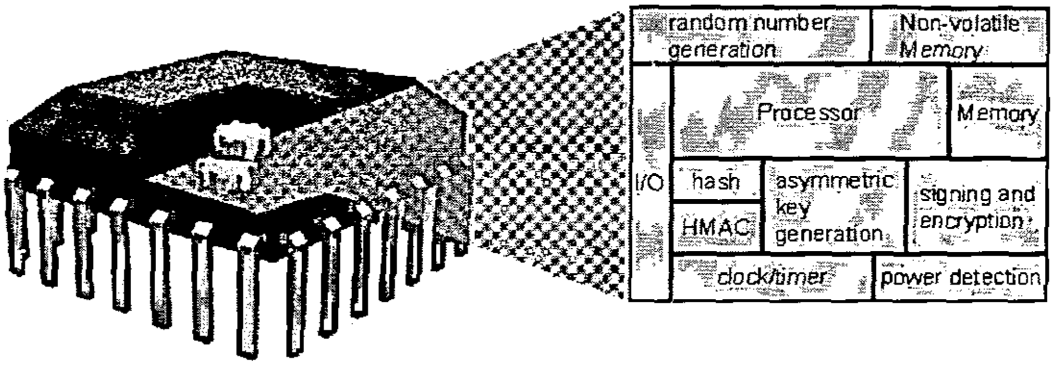


Figure 1.4: Trusted Platform Module [5]

The TFM, is equipped with special purpose registers called Platform Configuration Registers (PCRs). Each PCR can store 160-bit hash value and at each reboot, the PCRs are reset. Each software component of the platform is mapped to 160-bit hash value that represents its good state. Each new hash is concatenated with an existing hash in the PCR and the new hash is stored in the same PCR. In this way, small number of PCRs can be used to measure all the components of a platform. Further, a Stored Measurement Log (SML) is used to keep track of the sequence of measurements in the PCRs. Remote attestation is a mechanism in which a platform presents the PCR measurements within its TPM to a challenger. In a typical remote attestation scenario, a platforms TPM collects the requested PCR values with their indices, signs them with AIK and returns them to the challenger along with the SML. Using SML and PCR measurement values, it is possible for a challenger to re-compute the PCR values and compare them with their expected values. Based on the comparison, challenger can make a decision whether the target platform is in a valid state or not. *A* comprehensive introduction to remote attestation and trusted computing can be found at [4].

The TFM has been designed to protect security by ensuring the following:

- Private keys cannot be stolen or given away.
- Modification in the software code is always detected.
- Private keys are protected from being used by malicious code
- Sealing keys are not easily available to a physical thief.

The TCG chip accomplishes these goals with three main groups of functions, as follows:

- Public key authentication mechanism
- Measuring software integrity
- Attestation functions

1.1.3 SAML (Security Assertion Markup Language

SAML [6] is an XML-based framework for forming "security assertions", and exchanging them between entities. SAML assertions may be communicated through SAML protocol or other protocols to convey an assertion itself, or to communicate about the "subject" of an assertion.

1.2 Problem Domain

In the existing web access control and communication paradigm there is no mechanism to check the integrity status of the client before releasing any web resource. PCs connected to the Internet are exposed to a wide variety of threats. The protection against these threats can be provided by ensuring that these systems have required software and hardware configurations and having no Trojan or virus running on it. If a system is found to having rootkit or virus software that is not intended to be run on that system, it is considered to he compromised. Remote attestation provides the mechanism to detect this fact. Software-based solutions to security, such as anti-virus and firewall solutions, depend on the security of the underlying operating system at which they run. It is assumed that the operating system has not been tempered with. Firewalls and antviruses also store their secrets on the same system that they want to protect. These secrets are vulnerable for alteration and observation. Similarly, if the OS has been compromised, the security software can either be bypassed altogether, or it can be fooled into believing that a compromised system is actually secure. Moreover, the security software itself is vulnerable to attacks and malicious modification by the host on which it is running.

While the web server, before allowing access to the requested resource, did identify and authenticate the relying party to ensure its authenticity, the communication may also he taking place over Secure Socket Layer (SSL)[7] to ensure confidentiality, however, what it does not know is the integrity status of the machine on the other side. It could be possible that the machine on the other side of communication channel has been compromised because of some malicious software running on the platform and this could result in the resource falling in the hands of a malicious user.

1.3 Proposed Architecture

Clients' integrity measurement typically involves the use of special client software running on the system that observes and reports the configuration of the system to the requested system. While TPM is installed on a system we can make the client platform verifiable. TPM is equipped with Platform Configuration Registers (PCRs). The PRC can store the measurement of each software component loaded into the memory. Similarly, a Stored Measurement Log (SML) is used to keep track of the sequence of measurements in the PCRs. We propose the incorporation of remote attestation in the web communication paradigm. In this proposal the web server before releasing a protected resource can verify the client status. For this purpose we added an attestation service to the web server that verifies the software stack running on the client

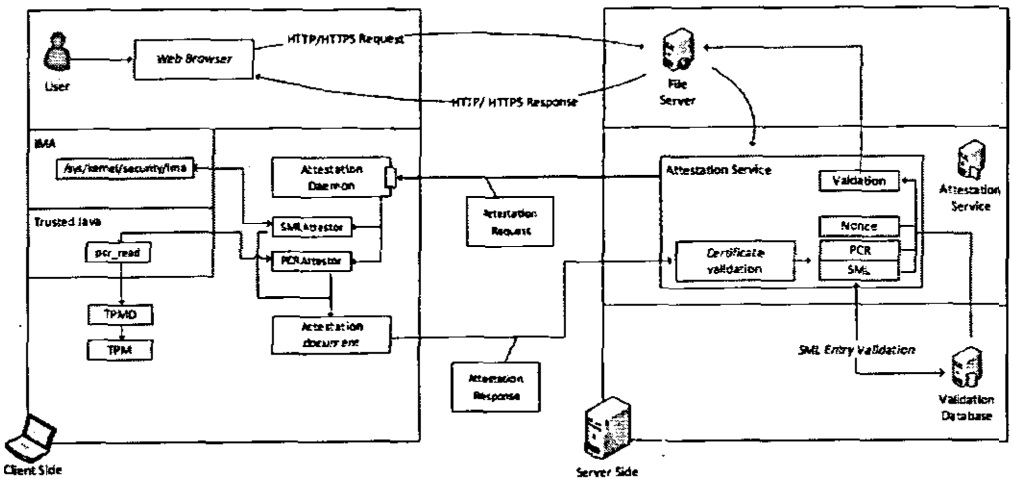


Figure 1.5 Proposed System

In other words we add integrity based access control mechanism in the web servers. This is a mechanism in which platform presents PCR measurements within its TPM to the challenger along with SML. Using SML and PCR measurement values, it is possible for the attestation service to re-compute the PCR values and compare them with their expected values. Based on the comparison, the attestation service can make a decision whether the target platform is in a valid state or not; based on which the protected resource is released to the client. The detailed procedures of the proposed system are discussed in chapter 5.

1.4 Thesis Outline

The rest of the thesis is organized as follows:

Chapter **2** describes the state of the art review of the existing remote attestation techniques and research in the area of web security.

Chapter **3** discusses the requirement of the inclusion of remote attestation in web servers and also discusses the issues that arise in this process.

Chapter 4 describes our proposed system architecture and its functional details

Chapter 5 illustrates the implementation of the prototype system with open source browser and web server.

Chapter 6 discuses the security evaluation of the proposed system and also describe some performance results and future work.

2. Literature Survey

2. Literature Survey

Remote attestation is a relatively new field of research. A lot of literature on system security, trusted computing and Integrity measurement and validation is available. The reviewed literature is divided into three categories, remote attestation, client side security and existing web access control and security mechanisms. Some of the recent research is reviewed as follows; literature survey is divided into following broad categories:

- Remote Attestation
- Client Side Security
- Web Access Control

2.1 Remote Attestation

Schoen (2003) in "Trusted Computing: Promise and risk" shows that Trusted computing initiatives propose to solve some of today's security problems through hardware changes to the personal computer. There is a strong debate on the trusted computing controversies that are going on for the past few years in the industry. The paper has some strong arguments in favor of Trusted Computing [8].

Rick et al. (2003) in "Establishing the Genuinity of Remote Computer Systems" the authors discussed the fundamental problem in distributed computing environments that involves determining whether a remote computer system can be trusted to autonomously access secure resources via a network. They described a method by which a remote computer system can be challenged to demonstrate that it is genuine and trustworthy. Upon passing a test, the client is granted access to distributed resources and served as a general-purpose host for distributed computation as long as it remains in contact with some certifying authority. This research provides good understanding of remote attestation through cryptographic techniques using software mechanisms. The paper covers general purpose application for genuinity checking while it does not address the whole system state and neither uses any hardware component for storing keys. Thus the system is vulnerable to software attacks [9].

Sailer et al. (2004) in "Design and Implementation of a TCG-based Integrity Measurement Architecture" presents the design and implementation of a secure integrity measurement system for Linux. All executable content that are loaded onto the Linux system, are measured before execution and these measurements are protected by the Trusted Platform Module (TPM) which is part of the Trusted Computing Group (TCG) standards. This system was the first to extend the TCG trust measurement concepts to dynamic executable content from the BIOS all the way up into the application layer. In fact they have shown that many of the Microsoft NGSCB[19] guarantees can be obtained on today's hardware and software. These guarantees do not require a new CPU mode or operating system but merely depend on the availability of an independent trusted entity e.g. TPM. They show how their system can detect undesirable invocations, such as rootkit programs, and that the measurement architecture is practical in terms of the number of measurements taken. In IMA, a target platform presents the trusted status (i.e. binary hashes) of all its components loaded after booting. However, IMA alone is not very practical in open and heterogeneous environments. IMA being very effective integrity measurement system for Linux platform, we chose IMA for integrity measurement in our proposed architecture [10].

Yoshihama et. al. (2004) proposed an architecture and implementation called Trusted Platform on Demand (TPod), which increases the trustworthiness of networked platforms by combining dedicated security hardware, a secure operating system kernel and an open security protocol, to provide a secure software platform that may host a diverse range of distributed applications. Especially significant is the fact that the applications are better protected even if there are vulnerabilities in the application software or in the system software, or even if the system administrator is not completely trustworthy. The architecture gives an understanding of maintaining and checking of platform integrity through dedicated security hardware. Trusted Platform on Demand (TPoD) implements a trusted boot sequence by attesting to BIOS and boot loader e.g. GRUB integrity. The Integrity Measurement Architecture (IMA) extends the trust chain established by TPoD by attesting to the load-time integrity of the Linux OS and its applications. TPoD proposes a generic system that incorporates remote attestation in any distributed system[11].

Sadeghai et. al. (2004) pointed out the deficiencies of the attestation and sealing functionalities proposed by the existing specification of the TCG. They argued that these mechanisms can be misused to discriminate certain platforms, i.e., their operating systems and consequently the corresponding vendors. They pointed out the problem of managing the multitude of possible configurations. They discussed how their approach overcomes these problems by proposing a new approach in which the attestation of a platform should not depend on the specific software and/or hardware but only on the properties that the platform offers. The property-based attestation only verifies whether these properties are sufficient to fulfill certain security requirements of the challenger who asks for attestation. They proposed and discussed a variety of solutions based on the existing Trusted Computing (TC) functionality. They demonstrated how a property-based attestation can be realized based on the existing TC hardware such as a Trusted Platform Module (TPM). The proposed approach is quite interesting but the problem comes when we need to identify the diverse range of configurations and then mapping them to specific properties. Moreover the technique does not provide root of trust concept has been proposed by the TCG compliant protocols[12].

Shi et. al. (2005) proposed BIND (Binding Instructions aNd Data), fine-grained attestation service for securing distributed systems. They propose the concept of code attestation. Existing code attestation technology is relatively immature due to the variabilities in software versions and configurations; verification of the hash is difficult. They identified time-of-use and time-of-attestation problem that remained to be there, since the code may be correct at the time of the attestation, but it may be compromised by the time of use. Their goal was to address these issues and make code attestation more usable in securing distributed systems. BIND claimed to offer a fine-grained attestation i.e. instead of attesting the entire memory content, it attests only the piece of code that challenger is concerned about which greatly simplifies verification. This narrowed the gap between time-of-attestation and time-of-use. BIND measures a piece of code immediately before it is executed and uses a sand-boxing mechanism to protect the execution of the attested code. Similarly it tied the code attestation with the data that the code produces, such that challenger can pinpoint what code has been run to generate that produced results [13].

Li et. al. (2006) identify a few fatal deficiencies to be overcome, e.g., leakage of platform configuration privacy or hard to define trustworthiness related properties etc.. To address these issues, they presented a system behavior based attestation model to determine the trusted state of attesting platform from its system trustworthiness related behaviors. The proposed attestation model claimed to have advantages of privacy protection and high feasibility. They were the pioneers to propose system behavior based attestation that can also be used to effectively constrain impacts caused by malicious code such as Trojan and virus which are common in today's organization business systems. Although they could not prove their concept to be capturing the dynamic behavior of the running system but opened a new concept of behavioral attestation which is a new challenge in today's remote attestation techniques [14].

Safford (2005) in "Trusted Linux Client" presents the project of Trusted Linux Client (TLC) which was to protect desktop and mobile Linux clients from on-line and offline integrity attacks, while remaining transparent to the end user. This is accomplished with a combination of a Trusted Platform Module (TPM) security chip, verification of extensible trust characteristics, including digital signatures, for all files, authenticated extended attributes for trusted storage of the resultant file security meta data, and a simple integrity oriented Mandatory Access Control (MAC) enforcement module. He has shown how one can make Linux as a trusted system. The paper describes new technologies that are deployed on the Linux operating system to make the platform trusted [15].

Yoshihama et. al. (2008) proposed WS-Attestation architecture that is built on top of the Web Services framework. WS-Attestation claims to be a software oriented, dynamic and finer-granular attestation mechanism which provides TCG and WS-Security technologies to increase trust and confidence in integrity reporting. It also allows binding of attestation with application context, as well as infrastructural support for attestation validation.
The authors states that

> *"If a user trusts an on-line shopping service and submits his credit card number even if the service provider company is actually honest and trustworthy the server platform might be infected with a Trojan horse that surreptitiously sends the credit card number to a malicious remote attacker. In another case, the server software might have a vulnerability that would be attacked by an attacker, allowing the attacker to obtain the super user privilege and steal customer information and credit card numbers. Therefore, it is important to make sure that the service is running on a trustworthy platform; i.e. it is running on the hardware that it claims to be, and that the OS and software are not infected by malicious software and has no known vulnerabilities".*

In WS-Attestation, the mechanism for the definition of a benchmark for the trustworthiness of a platform is not specified. It provides only a structural paradigm and coarse-grained framework on which more fine-grained attestation techniques can be deployed. Similarly it aimed to provide a software oriented, dynamic and fine-grained attestation mechanism that leveraged TCG and WS-Security technologies to increase trust and confidence in integrity reporting [16].

2.2 Client Side Security

Zishuang et. al. (2005) in "Trusted Paths for Browser" discusses the existing security mechanism for securing the web communication that is SSL (Secure Socket Layer) and also tells about the ways how they can be exploited. They showed how malicious servers can still do this-and can also forge the existence of an SSL session and the contents of the alleged server certificate. They discuss how to systematically defend against Web spoofing, by creating a trusted path from the browser to the human user and present potential designs, propose a new one, prototype it in open source Mozilla, and demonstrate its effectiveness via user studies. This is a very comprehensive research for understanding the existing problems in the web communication. However this paper emphasizes on the channel security and try to make the communication more authentic and secure, the paper does not address the system security [17].

Trusted Network Connect (2005) is the initiative taken by the TCG to incorporate remote attestation in the network layer protocols such as IPSec. TNC is an ongoing project of the TCG for which we do not have any practical implementation available as yet. The problem with TNC is that it works on the network layer which means that either total access will be granted to a client for entering in a network or denied altogether. There is no discrimination for the upper layer protocols in the TNC. We do not require web servers to be closed altogether. For web access control mechanism we need to mention some part of the web content to be protected and the rest being open for public access[18].

2.3 Web Access Control

Cryptographic protocols such as SSL[7], or those used in virtual private networks (VPNs), provide end-to-end authentication, but say nothing about the platform on either end or its integrity. For example, it is possible for a malicious or compromised client to initiate an SSL connection using a web browser the SSL connection only encrypts network traffic. Consider a corporate employee connecting to his company's network from outside it using VPN software. The VPN software authenticates the employee, but cannot check any of the properties of the particular client system with which the connection is being made.

2.4 Limitations

The client system may be compromised, or may leak company secrets once they are downloaded on it. Cryptography alone is an incomplete solution to security because it does not say anything about the behavior of the system.

Servers are attacked indirectly through compromised clients. While it is hard to directly take confidential data from a corporate server, it is much easier to compromise a laptop belonging to a user who has the necessary credentials, such as a password or client certificate, to get confidential data, and then steal the data from the laptop using malicious software. Remote attestation has been a hot area of research for the last several years but incorporation of remote attestation in the web servers has been an ignored since its birth. The reviewed articles do address the issues of remote attestation but no research has still been related to the web servers integrity based access control. So there is a need for the development of a mechanism through which remote attestation can be incorporated in the web server.

3. Requirement Analysis

3. Requirement Analysis

Before providing solutions for improved security for offering online services we need to identify the basic components of the service structure. The three basic components of any online service are server or service provider, communication channel or protocol and the consumer of the service i.e. client itself. All these components are one way or the other vulnerable to attacks. While keeping the domain of web communication we can identify these three components as:

- Web server
- HTTP/HTTPS being the communication protocol
- Client machine on which the browser is running

There has been discussed in the previous chapters there has been much emphasis on the server and channel securities. In order to provide improved security for web communication web servers should have access control mechanism based on security status of the client i.e. it can assess the security status of the client before releasing any confidential resource. In today's Internet the most vulnerable part of the online communication is the client [2]. That is why the need of remote attestation in the web servers will be quite effective.

3.1 Problem Scenario

Enterprise web applications still have reservations regarding remote working, and a major concern among them is security. Making the appropriate data available, but only to the right people; avoiding data leaks; securing digital assets and remaining compliant with medical, financial or corporate regulations; managing and supporting remote non-technical workforce are frequently cited as reasons against widespread adoption of remote worker programs. Cryptographic protocols such as SSL [7], or those used in Virtual Private Networks (VPNs), provide end-to-end authentication, but say nothing about the platform integrity. For example, it is possible for a malicious or compromised client to initiate an SSL connection using a web browser the SSL connection only encrypts network traffic. Consider a corporate employee connecting to his/her company's network from outside using VPN software. The VPN software authenticates the employee, but cannot check any of the properties of that particular client with which the connection is being made. The client system may be compromised, or may leak company secrets once they are downloaded on it. Once again, cryptography alone is an incomplete solution to security because it does not say anything about the behavior of the system. There are scenarios where a compromised client takes the protected resource and because of the malicious software data can be taken the client using back door software. For example an employee downloads company's confidential web page or any other document to his system and some rootkit running on the client send that document to the hacker through email. Similarly, servers can be attacked indirectly through compromised clients. While it is hard to directly take confidential data from a corporate server, it is much easier to compromise a laptop belonging to a user who has necessary credentials, such as a password or client certificate, to get confidential data, and then steal the data from the laptop using malicious software.

3.2 *Requirements*

The proposed system has the following design requirements

3.2.1 Flexibility

The inclusion of an attestation module to the web server must be flexible enough to support remote attestation techniques. The architecture should be modular enough to incorporate different attestation techniques based on the organizational requirements. Similarly it should be able to support different attestation models so that a validation service can also be included in the architecture.

3.2.2 Measurement Correctness

Collection of the integrity information should be verifiable so that the challenger can verify that the collected information is indeed taken from the system and is signed by a genuine TPM.

3.2.3 Dynamicity

Trusted boot measures the integrity of all executable components loaded uptill the operating system. Different components and data loaded in the operating system and application affect the behavior of a running system. For trust-aware web server it is important to support rich semantic attestation information.

3.2.4 Integrity verification

There should be an affective detection mechanism of malicious software. For verification of the individual application there should be a database of known good and known bad hashes. The database will be queried for each SML entry so that integrity of each software component running on the client can be verified.

3.3. Focus of the research

The requirements for the integrity based access control mechanism for a web server are:

i. The web server should be able to specify the protected resources

ii. It needs to have mechanism to ask for the trusted status of the users that are working at a remote site before allowing access to a protected resource

iii. It should have the mechanism to validate client platform integrity status

iv. Based on the integrity status the appropriate resource of error code should be sent to client

Similarly the client platform also has the requirements as follows:

1. The browser while accessing a protected resource should be able to validate the validation request
2. The browser should be able to communicate its platform status in a secure and authentic manner.

4. System Design

4. System Design

Limited access to Web pages can be implemented through PKI certificates and client side SSL connections. Access can be limited to holders of PKI certificates issued by a particular organization or even by specific individuals. In such an SSL connection, the web server specifies that the user must open their end of the connection with a personal private key or public certificate. Once the client side SSL connection is established, the web server allows access to the contents. This connection in turn uses this information to authenticate the user and decide whether to authorize access to the application. Setting up the server side of the application obviously depends on the web server software being used. A web server has three distinct ways of dealing with the question of whether a particular request for a resource will result in that resource actually be returned. These are called Authorization and Access control. Authentication is by which we can verify that someone is who they claim to be. Normally in web servers it is done through username and password.

Authorization is finding out if the user is permitted to have access to a particular resource. This is usually determined by finding out if that person is a part of a particular group or has a particular level of security clearance. Access control is a more general way of talking about controlling access to a web resource. Access can be granted or denied based on a wide variety of criteria, such as the network address of the client, the browser which the visitor is using or the time of day. Access control is controlling entrance by some arbitrary condition which have anything to do with the attributes of a particular visitor. Although these techniques are different in their concepts still they are closely related in most real applications, it is some times difficult to talk about them separatly from each other. We are making some part of our website that has some sensitive information, protected. Based on the integrity status of the client the pages will be visible to in the browser. We are adding a module to the web server so that it can verify the integrity status of the client before releasing a protected object to the browser. Similarly on the client side secure integrity reporting mechanism is deployed that enables the browser to send the integrity information to the server on request.

4.1 Design Requirements

To incorporate remote attestation for integrity based access control in the web server, we need to know the attestation models that need to be incorporated in the proposed architecture. In order to support flexible communication between service requester (SR), service provider (SP) and a validation service (VS), the WS-Attestation proposes three different models.

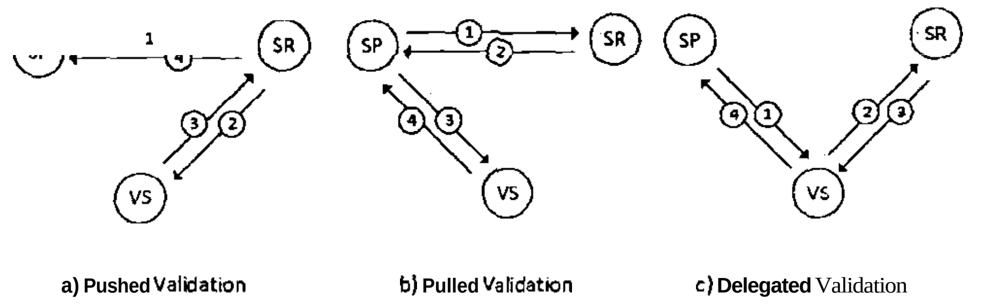


Figure 4.1: Attestation Models

In Pushed Model, a requester takes an attestation credential in advance and presents it to the service provider. In Pulled Model, the service provider requests the validation service to prove the platform integrity of a requester. Finally, in Delegated Model, the service provider requests the validation service to perform the attestation on its behalf. However, in the proposed models of WS-Attestation, the mechanism for the definition of a benchmark for the trustworthiness of a platform is not specified. More specifically, it provides only a structural paradigm and coarse-grained framework on which more fine-grained attestation techniques can be deployed. In the web server we incorporated delegated model for validating the client's platform integrity.

4.3 Operational Details

We are making some part of our website that has some sensitive information, protected. Based on the integrity status of the client the pages will be visible to in the browser. We are adding a module to the web server so that it can verify the integrity status of the client before releasing a protected object to the browser. Similarly on the client side secure integrity reporting mechanism is deployed that enables the browser to send the integrity information to the server on request.

Figure 4.2 depicts the proposed architecture in detail. On the web server, there is a distinction between open and restricted content. The means of restrictions on content is done through certain access control mechanisms. Our access control mechanism is termed as Integrity Based Access Control (IBAC). IBAC on a web server is a novel idea which has not been studied elsewhere to date. The steps in Figure 4.2 describe the request and response process in our architecture. In order to facilitate this functionality on a web server, the server should have a module of integrity verification for attesting the client. For this purpose, we propose an attestation service that performs this function on behalf of the web server. The modular structure of our proposed architecture supports the three architectural models of WS-Attestation. The request handler in the web server communicates with the attestation module through the Security Assertion Markup Language (SAML) [6] protocol. This ensures future scalability of our architecture into web services.

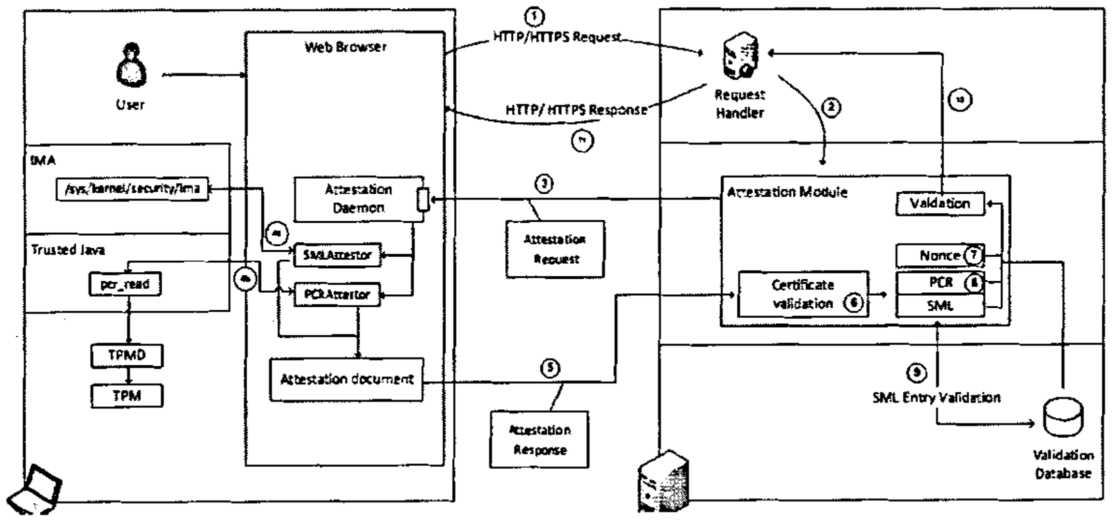


Figure 4.2: Proposed Architecture

In Step 1, the client requests for a protected object on the web server. The request handler at the web server end, after recognizing the target as being restricted, passes control to the attestation module (Step 2). The attestation module makes an attestation request to the client which includes a `SignChallenge` i.e. a nonce (Step 3). The attestation daemon on the client performs steps to collect integrity information of the client platform. In Step 4a, it collects SML from the *securityfs*. Similarly, in Step 4b, it passes the received nonce to the Trusted Software Stack (TSS) [25] along with a request for a *TPM_QUOTE*. The SML and the quote are aggregated in an attestation response document and returned to the web server. HTTP Response The attestation module at the web server has three basic functionalities for verification of this information: 1) Nonce verification: for ensuring that the response received from the client is fresh (Step 7), 2) PCR verification: for ensuring that the nonce and SML are indeed signed by a genuine TPM and are not reported by a software application masquerading to be a TPM(Step 8) and 3) SML verification: for ensuring that no rootkits or other malicious software is running on the client (Step 9).

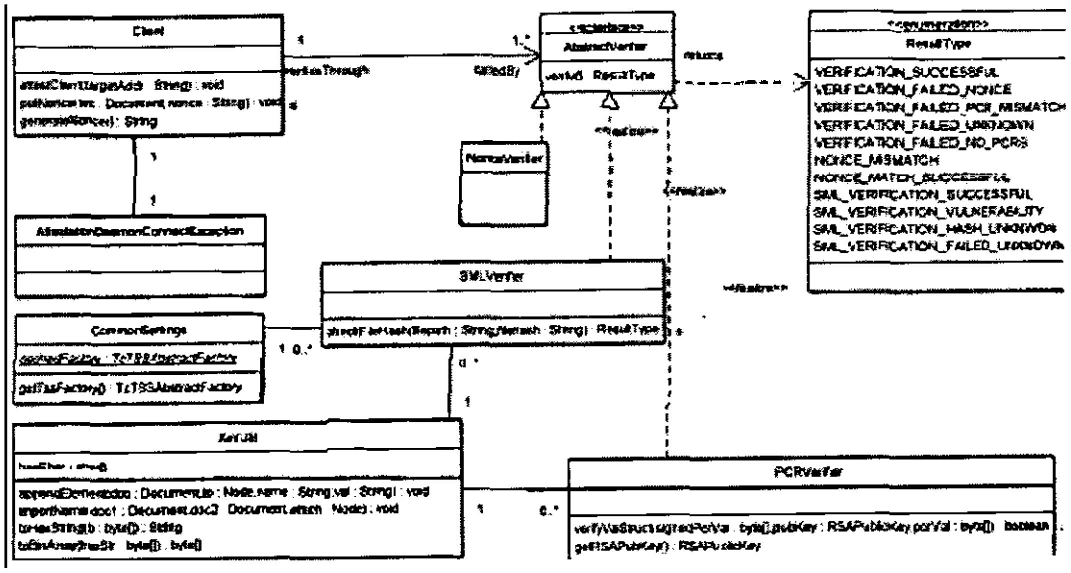
For SML verification, there must be a validation database which validates the hashes returned by the client for their corresponding executables. This validation database can

TH. 9986

be constructed through collecting individual known good hashes of all the executables on a target platform. The following section details the classes to be developed for the implementation of the attestation information collection, transmission and validation.

4.4 Class Diagrams

The public interface for the attestation engine is the `AttestationModule` class. The class should expose a public function called `AttestClient` which takes a single target address parameter SAMLReq of type Document which is a SAML request describing the required attribute assertion for attestation and the client. The class internally implements two private functions generate nonce and put nonce which create and insert a unique Nonce in the attestation request document. an Enumeration of result **type** is created to describe the verification results.



Client
attestClient(targetAddr : String) : void
putNonce(req : Document, nonce : String) : void
generateNonce() : String
verifiesThrough
1
1..*
AbstractVerifier
verify() : ResultType
returns
NonceVerifier
«realize»
AttestationDaemonConnectException
CommonSettings
getTssFactory() : TcTSSAbstractFactory
SMLVerifier
1 0..*
1
0..*
KeyUtil
PCRVerifier
getRSAPubKey() : RSAPublicKey
«enumeration»
ResultType
VERIFICATION_SUCCESSFUL
VERIFICATION_FAILED_NONCE
VERIFICATION_FAILED_PCR_MISMATCH
VERIFICATION_FAILED_UNKNOWN
VERIFICATION_FAILED_NO_PCRS
NONCE_MISMATCH
SML_VERIFICATION_SUCCESSFUL
SML_VERIFICATION_VULNERABILITY
SML_VERIFICATION_HASH_UNKNOWN

Figure *4.3:* Class diagram of Attestation client daemon

- *VERIFICATIONSUCCESFUL:* validation of the PCR signature value succeeded.
- *VERIFICATION_FAILED_NONCE*: The nonce was not returned by the client

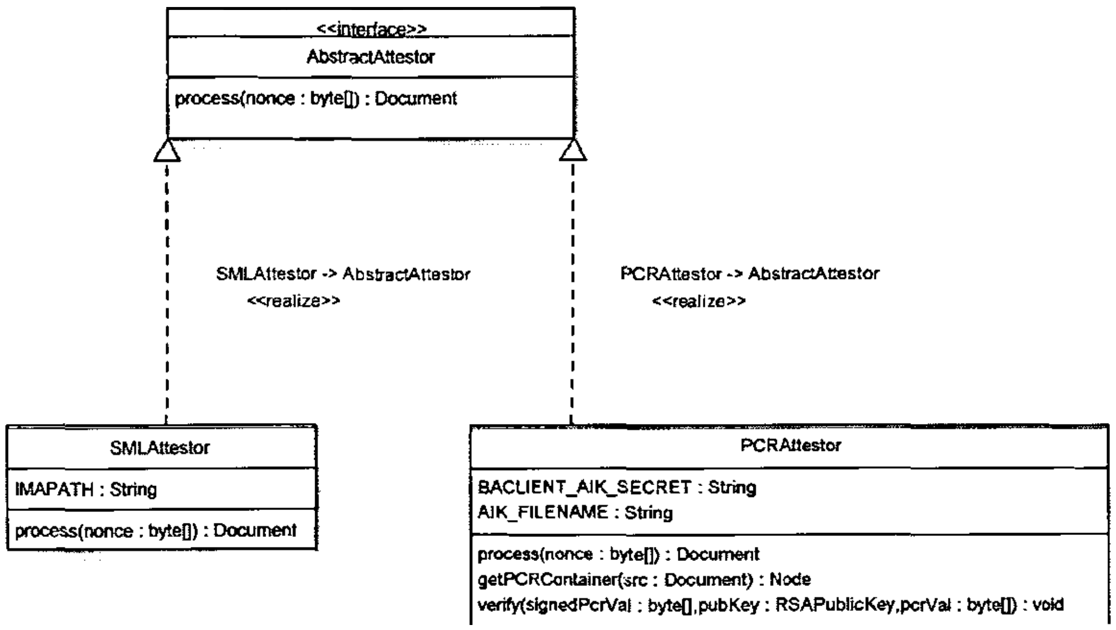
- *VERIFICATION_FAILED–PCR–MISMATCH:* PCR signature validation failed
- *VERIFICATION_FAILED_UNKNOWN*: validation of the PCRs failed for an unknown reason. This might be due to the unavailability of the client's public key or inability of a signature validation etc.
- VERIFICATION–FAILED–NO–PCRS: No PCRs were returned by the client
- *NONCE_MISMATCH*: The return nonce did not matched
- *NONCE_MATCH_SUCCESSFUL:* The nonce match succeeded. The nonce sent is match against the TPM signed nonce to assure freshness.
- *SML_VERIFICATION_SUCCESSFUL*: The aggregate of the SML matched with the PCR values sent by the client.
- *SML_VERIFICATION_VULNERABILITY* : A vulnerability was found in the SML
- *SML_VERIFICATION_HASHUNKNOWN*: An unknown hash was encountered in the SML
- *SML_VERIFICATION_FAILED_UNKNOWN:* Failed for an unknown reason.

In order to provide the functionality of a pluggable architecture for incorporating new attestation techniques in the future we require an interface **AbstractVerifier** which exposes a public function verify which can take any type of attestation credentials. In the current scenario we need three realizations of this interface.

1. *NonceVerifier*: Verifies that the nonce sent to the client was signed by the TPM and returned in the attestation response document. This assures the freshness of the nonce.

2. *PCRVerifier* : Verifies that the **PCRComposite** Structure was indeed signed by the AIK of a valid TPM. This provide assurance in the correctness of the PCR value.

3. *SMLVerifier* : Verifies that all the hashes in the SML correspond to some known good hashes to assure that no rootkits or other malicious application/software are running.



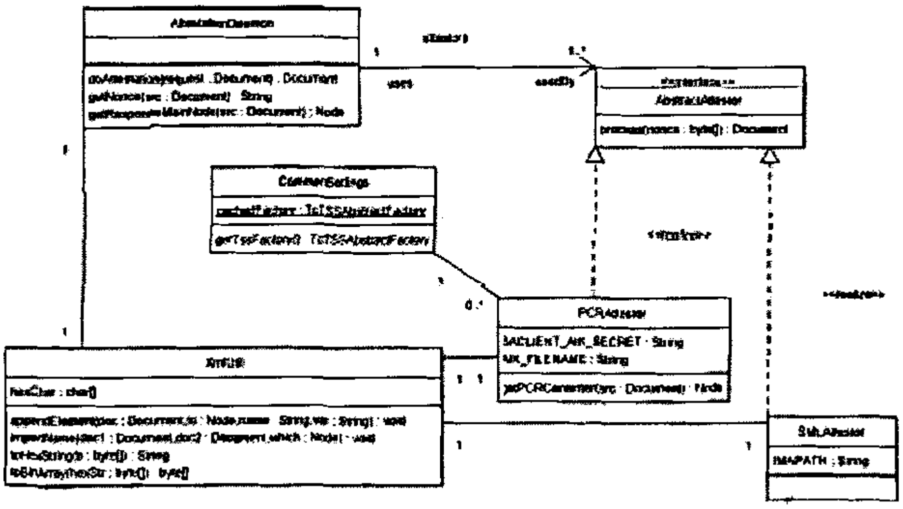
<<interface>>
AbstractAttestor
process(nonce : byte[]) : Document
SMLAttestor -> AbstractAttestor
<<realize>>
PCRAttestor -> AbstractAttestor
<<realize>>
SMLAttestor
IMAPATH : String
process(nonce : byte[]) : Document
PCRAttestor
BACLIENT_AIK_SECRET : String
AIK_FILENAME : String
process(nonce : byte[]) : Document
getPCRContainer(src : Document) : Node
verify(signedPcrVal : byte[],pubKey : RSAPublicKey,pcrVal : byte[]) : void

Figure 4.4: Abstract Attester Class

Two helper classes `CommonSettings` and `XmlUtil` are defined to provide certain functionalities of the TSS and XML manipulation respectively. We have utilized this exibility for creating two functions related to remote attestation. 1) `MeasurementCorrectness` and **2)** `SoftwareIntegrity`. The validation module utilizes these two logical functions for the purpose of attestation.

The TPM takes the current value of PCR-10 and computes a `PcrComposite` structure. Fig **4.3.** shows the internal representation of the `PcrComposite` structure. The TPM then computes a SHA-I hash of the structure. The hash is then appended to some fixed values (as defined by the TPM specification) and the nonce passed to the TPM is then appended to this structure. The final value is a `TcBlobData` structure. The TPM then signs this structure using the AIK (bacaik). The resulting digital signature and the `TcBlobData` structure are submitted to the VS along with the SML.

It is the attestation daemon that is part of the browser and it collects information about the system and attests this information from the TPM and makes an XML WS-Attestation response for the web server. This response is sent to the server for verification of the client platform. The *AttestationDaemon* acts as a service and is implemented within the web browser to provide for seamless handling of attestation requests from the web server. The daemon provides a single public function **doAttestation** which takes, as input, an attestation request in the form of an XML document.



It includes two private functions **getNonce** and **getResponseMainNode** which act as helper functions for attestation. To provide for a pluggable architecture of attestation, the abstract interface called **AbstractAttestor** is used in this class. This interface exposes a single function process which takes as input a 20-byte nonce and returns a WS-Attestation response in the form of an XML document. In our implementation, we have implemented two attesters **PCRAttestor** and **SMLAttestor. PCRAttestor** provides a TPM-signed quote over the current values of the PCR and the nonce sent by the challenger. Internally, it uses an ***Attestation Identity Key*** (**BACLIENT** *AIK)* for signing the values of the PCR. The **SMLAttestor** is a simple implementation of the **AbstractAttestor** which returns the Stored Measurement Log (SML) that is retrieved from the **securityfs.** The combined result of all the attesters is returned to the challenger by the *AttestationDaemon* in the form of a WS-Attestation response document. The package also utilizes **XmlUtil** and **CommonSettings** classes for different helper functions of XML and TSS respectively.

5. Implementation

5. Implementation

We have developed a prototype of the delegated model discussed earlier. The prototype is developed by extending the open source java-based web server jetty [20]. We have constructed a validation module specifically for performing attestation of a Linux Fedora Core 6 system. The web server requests this validation module through SAML to perform validation on behalf of the server. The validation module after performing attestation of the client responds to the web server with a SAML response. Similarly on the client side we extended a java based web browser Lobo[21] to incorporate integrity reporting mechanism in the browser. Detailed discussion of the implementation is discussed in the following sections.

5.1 Development Environment

For implementation of the proposed solution we need to have a TPM equipped system. The client system was a Dell Optiplex 745 desktop computer having Atmel TPM chip onboard. Linux kernel 2.6 kernel automatically detects the TPM and installs its driver as a kernel module. tpm_tis kernel module is installed for communication with the TPM chip. We used Eclipse for with JDK 1.6 Eclipse is an open source project. Whose developers are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vivacious network of technology vendors, innovative start-ups, institutions, research community and individuals extend, complement and support the Eclipse platform. It is noted that the Eclipse Foundation does not actually develop the open source code. All of the open source software at Eclipse is developed by open source developers that are called committers, which are volunteered by individuals and organizations.

5.2 Trusted-Jetty

We implemented our prototype by changing the Jetty[20] web server which is an open source java based web server. We added our modules to the Jeny web server. The request handler of the Jeny web server was altered to incorporate integrity based access control mechanism in the web server. Trusted-Jeny is an open source project hosted at sourceforge [23]. The flow diagram shows the internal functionality of the web server for processing requests for integrity protected requests.

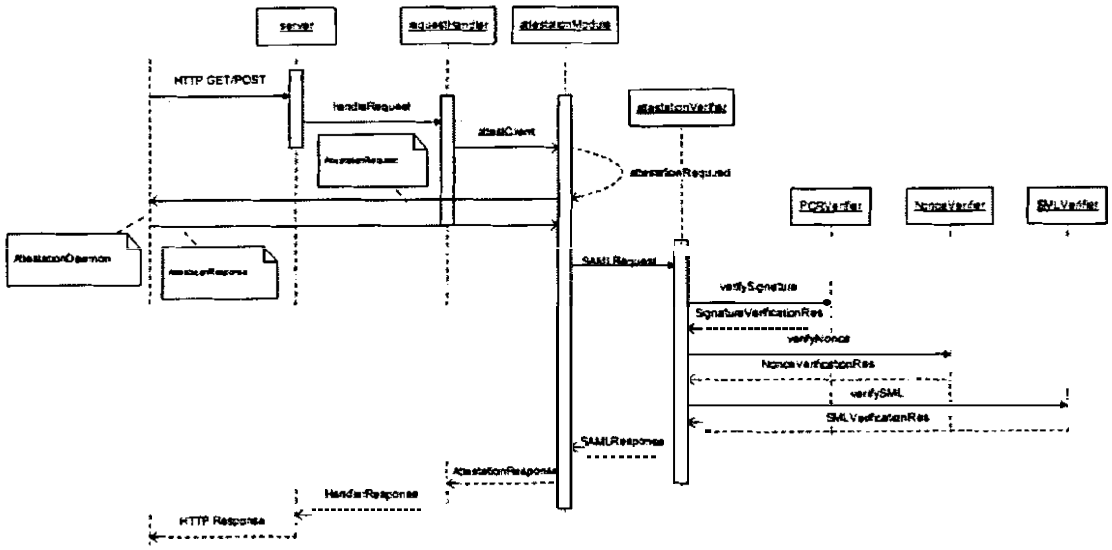


Figure 5.1: Sequence diagram of the server side

5.3 Trusted-Lobo

We have altered the java based open source browser Lobo [21]. We named our changed browser as Trusted-Lobo which is an open source project [24]. The sequence diagram at the client end depicted in the following figure

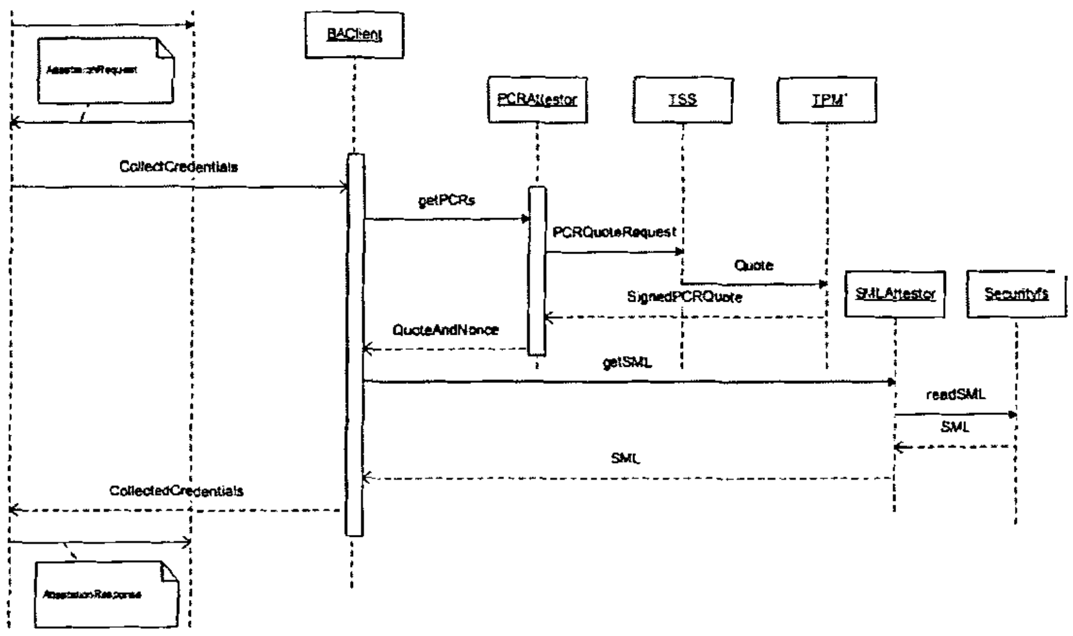


Figure 5.3: Sequence diagram at browser end

The validation module requests the client to return the Stored Measurement Log (SML) along with the current value of its tenth PCR, signed by the TPM. We have used Trusted Java [25] libraries for the incorporating attestation information acquisition in the web browser. Trusted Java provides a Java software stack (jTSS) for the purpose of communication.

The validation module asks the client for attestation along with a nonce (for ensuring freshness of the values returned). Upon receiving an attestation request from the validation module, `BAClient` running on the client reads the SML from /sys/kernel/security. It also asks the TPM to provide a quote over the value of it's tenth PCR. The TPM attests the values of its PCRs through the quote. The TPM takes the index of the PCR to quote, a nonce for assuring the freshness of the quote and an AIK for signing the value with. It appends the current PCR value with the nonce and digitally signs the result. This resulting structure is called a TPM quote over a PCR value.

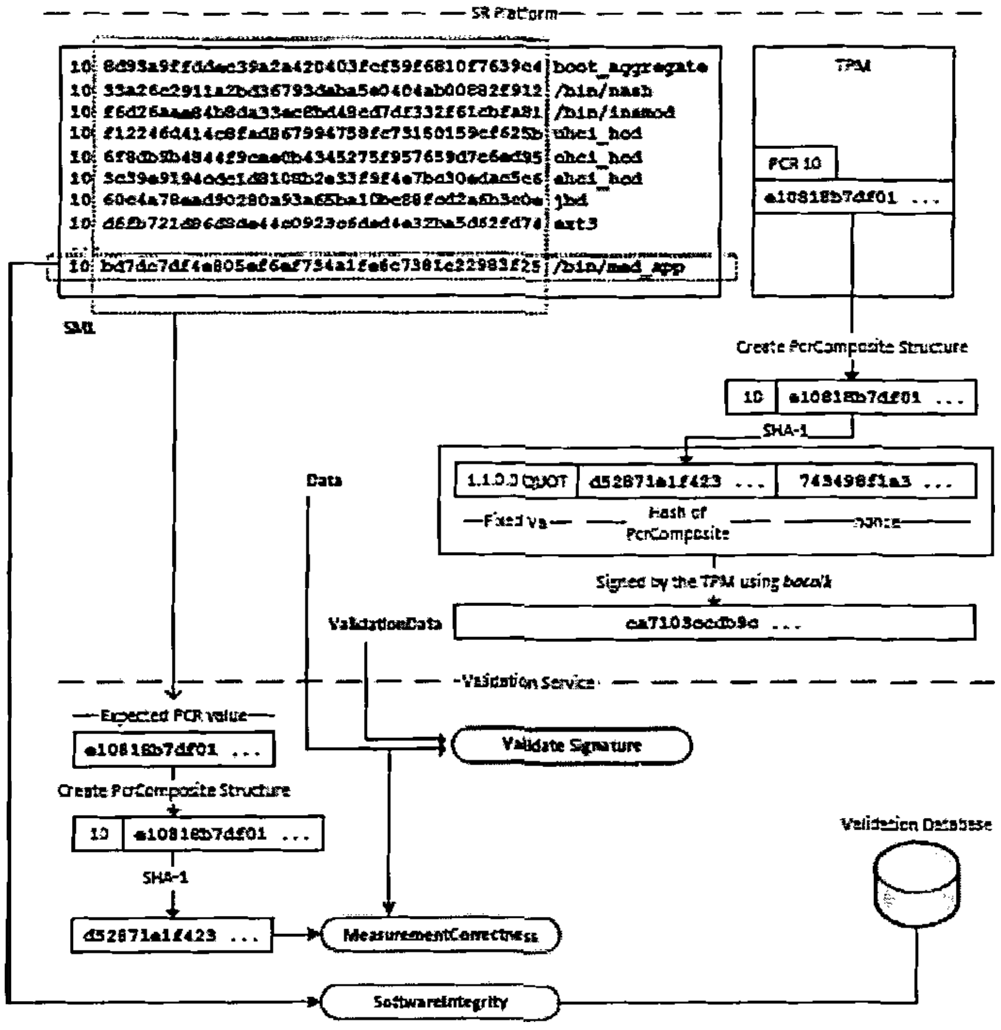


Figure 5.3: Operational details of the architecture

Upon receiving this quote from the client, the validation module first verifies the digital signature on the received quote through `PCRVerifier` to verify that it has indeed been signed by the TPM of the client. Then Validation module calls upon `NonceVerifier` to check the nonce for freshness. Afterwards, `SMLVerifier` is called upon to compute the expected value of the PCR is using the SML.

The algorithm for this computation is given below:

I. Compute the expected value of the 10th PCR value by accumulating the hashes in the SML starting from the initial PCR value of a 160-bit zero value:

PCR_{i+1} = SHA-1 (PCR_{i-1} || SML_i) where **i** = 0,......, n and n is the total number of hashes in the SML.

2. Compare the expected value of the PCR with the value returned by the client platform.
3. Compare the nonce returned by the client platform with the nonce sent to it.

If the expected values of the PCR and those returned by the client match and if the nonce is returned in the quote, the validation module can conclude that:

I) the SML sent by the client is indeed correct as it is signed by a valid TPM and

2) the values of the SML are fresh as the TPM has also signed the nonce while performing the quote.

For providing assurance of the integrity of all the application running on the client, we make `AttestationVerifier` class. It verifies the hash of the applications provided in the SML. Since the correctness and freshness of the software hash is assured by the TPM signature, the only requirement for establishing the integrity of the software is that the hash be of a known good value. For this purpose, we created a small database of known good hashes for all application. Since SHA-I produces an irreversible hash of the application. If the hash provided by the client corresponds to a known good value already present in the database, the validation module can certify the integrity of the system.

6. Results and Discussion

6. Results and Discussions

We need to evaluate our system in two categories. First one is the security evaluation and the second is evaluating the performance overhead caused by integrity validation steps in the web communication.

6.1 Security Evaluation

As the TPM is equipped with special purpose registers called Platform Configuration Registers (PCRs). One PCR can store 160-bit hash value and at each reboot, the PCRs are reset. Each executable of the platform is mapped to 160-bit hash value that represents its integrity state. Each new hash is concatenated with an existing hash in the PCR and the new hash is stored in the same PCR. Further, a Stored Measurement Log (SML) is used to keep track of the sequence of measurements in the PCRs. On validation request the client platform presents the PCR-10 value, signed by its TPM AIK, to the web server along with the SML. Using SML and PCR measurement values, it is possible for a challenger to re-compute the PCR values and compare them with their expected values. Based on the comparison, the web server makes a decision whether the target platform is in a valid state.

The system is in a trusted state if the PCR values are correct and fresh and the SML sent does not contain any vulnerable or unknown hash then we conclude that no malicious application is running on the target platform.

6.1.1 Security against malicious software

Assume that there is a malicious software running on the client machine. Since IMA (Integrity Measurement Architecture) is patched with the kernel of the client system, the hash of the malicious application will be recorded in the SML and aggregated in the value of PCR-10. Even if the operating system is compromised and it intends to fake the PCR values, it has to know the AIK used for signing the quote. Since the AIK, by dentition, never leaves the TPM [5], the operating system cannot alter the PCR value in a trusted manner.

Another way for the system to load an executable such that the result of thepcrextend function is a state before the malicious application was loaded. Since SHA-I is irreversible [26], such a hash cannot be computed feasibly. Hence, the application cannot restore the value of the PCR-10 to a past trusted state. Hence we conclude that it is not possible for a malicious application to hide itself after getting loaded.

6.1.2 Security against man in the middle attack

Assume that the PCR value or SML is altered in transit. The change can be detected while verifying the signature of the attestation response. If the PCR value is altered during transit then the change can be detected in signature verification. Similarly if the SML is compromised during transmission then the ***PCR Composite*** sent by the client and the one calculated during verification process will not match, which will result in an integrity violation error. Thus the PCR or SML cannot be altered in an un-detectable way.

6.1.3 Security against replay attack

Finally, the application might try a replay attack by sending old values of the PCR in response to an attestation request. The web server will send an un-predictable nonce into the attestation request. On reception of the nonce in request, it is embedded into the ***PCR Composite*** structure, which is signed by the TPM and is sent back to the server in response. The nonce can be taken out from the PCR Composite structure and matched with the one sent. If both matches it means that the response is fresh otherwise it will be considered as a replay attack. Thus nonce cannot be quitted in a trustworthy way.

6.2 Performance *Evaluation*

We have used a Dell Optiplex 745 system with TPM chip installed. The system has 1.7GHz processor and 256MB memory. A normal attestation time for a Red Hat Fedora Core 6 system installed with default options and running 5 to 7 desktop applications, takes about 14 seconds to complete attestation. The round trip time will of course be changed with the size of the resource and bandwidth. Figure 6.1 shows the variation of the attestation time with respect to system up-time.

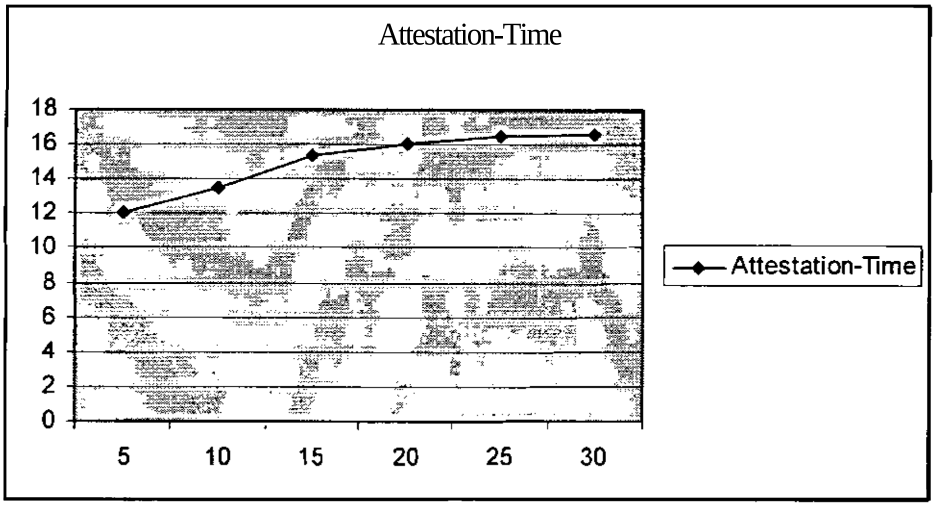


Figure 6.1: Attestation time variation vs. system uptime

As IMA (Integrity Measurement Architecture) sends stores the log of all the executables and libraries that are loaded at runtime, so the length of the SML increases with the system uptime and with the frequency of the loading applications. Our observed SML length with the passage of time is shown in figure 6.2.

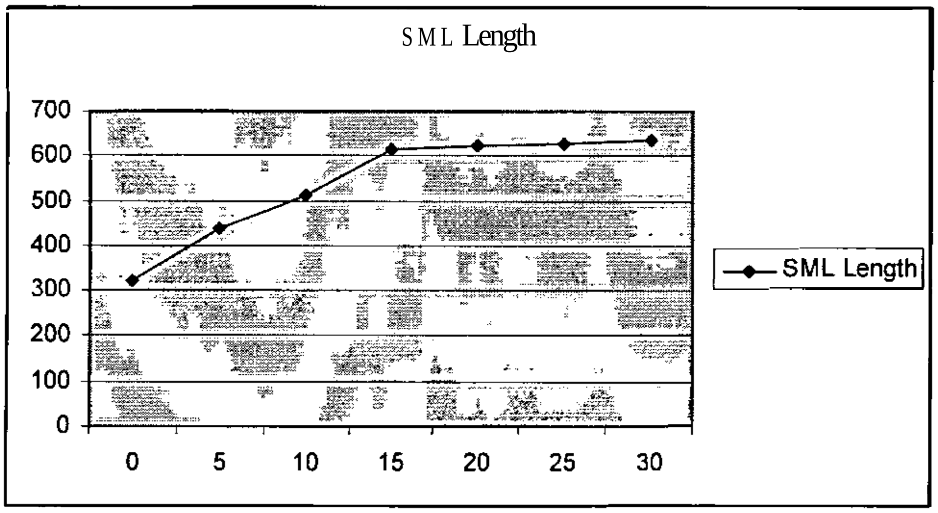


Figure 6.2: SML-Length w.r.t System Up-Time

In order to evaluate the attestation time we decompose full attestation time T_{fa} as follows:

$$T_{fa} = SAML_{req} + Att_{req} + PCR_{att} + SML,,, + Att_{res} + Cert_{ver} + PCR,, + SML_{ver} + SAML_{res}$$

We observed that the length of the SML increases with the system uptime. As the system is booted and no application program loaded into the system then the SML length is about 320 measurements.

Process	Description
$HTTP_{Req}$	HTTP request to arrive on the server
$SAML_{req}$	SAML request to the attestation module
Att-req	Attestation request to the client platform
PCR_{att}	Attestation of the PCR by the TPM
SML_{Att}	Taking the SML from the *securityfs* and incorporating it into the attestation response.
Att_{Res}	Attestation response to the web server
certificate-ver	Verifying the certificate
PCR_{Ver}	Verifying the nonce
SML_{Ver}	Verification of the SML entries against integrity database
$SAML_{Res}$	SAML Response to the request handler
$HTTP_{Res}$	HTTP response to the client browser

Table 6.1: Attestation process description

Figure 6.3 shows a decomposed attestation time taken with individual function time. The list of individual processes and their descriptions are detailed in table 6.1. The graph in figure 6.3 shows that the process time of PCR_{att} , SAMLreq, SAMLres and Attreq does not change. As the size of the SML does not affect these processes. While SMLver and Att-res processes execution time increases with the size of the SML. The time of SMLver can be optimized with the introduction of parallel query execution. Moreover sophisticated hardware can also improve the aforementioned parameters.

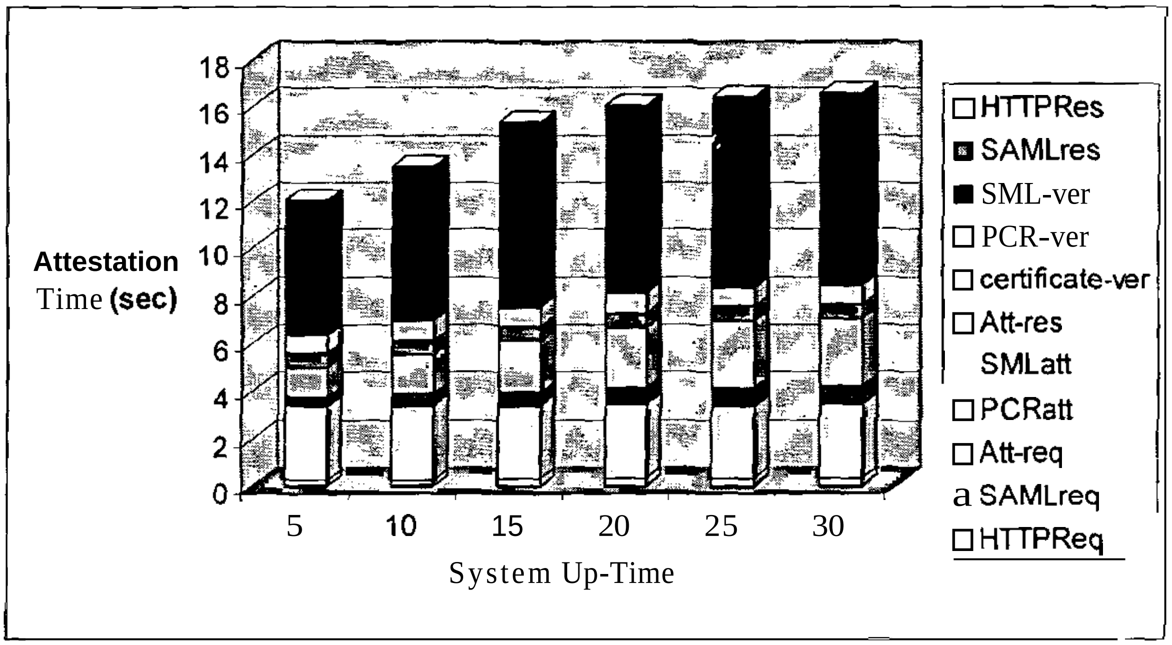


Figure 6.3: Decomposed Time chart for Attestation vs. System Up-Time

The attestation time varies with the running applications on the client system. The more applications run on the client machine the bigger will be the SML and accordingly verification time will also increase. Figure *6.2* shows the attestation time taken with respect to system up-time. By system uptime we mean that the system is in continuous use i.e. the user is running some normal desktop applications such as a web browser and word processing applications.

In our implementation round trip time including attestation and validation of client with 550 hashes took about 14 seconds as compared to WS-Attestation took about 1 minute for only 100 file hashes [16].

Attestation is done once the client accesses the protected resource as the web server maintain client session with it. On expiration of the session the new requests are verified for integrity attestation again. Attestation session can be set according to the security need of an organization. In a relaxed environment attestation time out can be set to longer duration as compared to more restricted environments where we may need attestation for each request. In order to avoid the problem of time of release and time of attestation we can set the session time to zero.

Attestation session avoid unnecessary attestations for each request. Once the platform is attested it can be trusted for that session. It is quite possible lower probability that the system gets compromised during the same session. We have set default session to 10 minutes, which is configurable by the administrator according to the need of the organization.

The reasons that the attestation takes longer time are the following:

- Only one operation can be performed at a time.
- It is really slow.
- It has a Finite number of resources, including key slots, auth slots, and so on.
- Communication to it is serial and via a driver.
- Communication to it is limited to local software.

TPM was designed to be relatively inexpensive in order to be adopted heavily. So a number of features that would have been nice to be embedded in the TPM were not included. These features were a real-time clock, a symmetric encryption engine, support for elliptic curves, and support for extended secure hash algorithms. All of these functionalities would have been nice to be included in the TPM. Some of them may appear in later versions of the specification [27].

6.3 *Conclusion*

In this research **we** incorporate remote attestation in the web server. Web server are enabled to enforce integrity based access control policies using Trusted Computing technologies which are supported and promoted by most of the industry giants in operating system and hardware industry.

We believe that the advent of trusted computing technology will change the way, how security aspects are envisioned while integrating business processes based on web-services. An effort [16] has been made to incorporate remote attestation in the web services architecture. However, these approaches are focused on communication between different entities involved in a remote attestation scenario at a higher level of abstraction. Remote attestation is currently a hot research area. In the current work we have incorporated binary attestation in the web server. Work is going on to improve the techniques of measuring, communicating and verifying the integrity of a remote platform. The proposed system is pluggable architecture and is based on the open protocols. Incorporation of a new attestation technique is matter of plugging a new implementation of the abstract interfaces in the proposed architecture.

6.4 Future Recommendations

Research is going on to address the limitations of the existing remote attestation techniques and to devise a new attestation mechanism that can measure the internal behavior of an application. There is multitude of future directions that we think can be opted. We opt some of them in our ongoing work. We identify two major research directions from this work.

Firstly, A new techniques need to be devised to measure the dynamic behavior of a remote computer. Binary attestation has some limitations. The major limitation is that it cannot measure the dynamic behavior of an executable. Property based attestation and configuration based attestations [12] were proposed in contrast to binary attestation which attests properties rather attesting binaries. Some specific configurations are being mapped to the specific properties to avoid leakage of the platform configurations. This is even harder to incorporate multitude of properties in a variety of software or system configurations. **A** new concept of remote attestation "Model-based Behavioral Attestation" [28] has been proposed. MBA is a high-level framework that abstracts the details of low-level attestation techniques. It is not a new attestation technique. Rather, it provides a framework through which the existing low-level techniques can be selected based on different scenarios and transformations. It generalizes behavioral attestation and associate behavior with policy models instead of individual security policies. Since the attestation is concerned with the behavior of a policy model, the attestation is more exact, simple and efficient. There is a margin in the realization of the MBA which is still at the model level.

Secondly, Extension to this work can be done by incorporating remote attestation into the shibboleth architecture [29]. Shibboleth is open source architecture for online secure authentication and SSO (Single Sign On). The concept of SP (Service Provider) can be mapped to the web server and IdP (Identity Provider) can be mapped to a validation service. In shibboleth user can select their identity provider for authentication; similarly, in the new architecture the user will be able to select their own validation servers to assure privacy and confidentiality. Shibboleth is already deployed and tested solution for web single sign on systems. The incorporation of remote attestation in Shibboleth protocol will enhance session management of our architecture.

References

References

[1] Symantec Internet Security Threat Report, 2008. http:Neval.symantec.coml mktginfo/ enterpiselwhite-papers1 b-whitepaper-internet -security-threat -report-xiii-04-2008.en-us.pdf.

[2] David Safford. "*Need for* TCPA ", March 2002

[3] Symantec Security Response (Silentbanker). hnp://www.symantec.com/ security-response/writeup.jsp? docid=2007-121718-1009-99.

[4] Trusted Computing Group (TCG). https:// www.trustedcomputinggroup.org/.

[5] Trusted Platform Module (TPM) Specifications. https:Nwww.trustedcomputing group.org/ specs/TPM/.

[6] Security Assertion Markup Language (SAML) v2.0. http://docs. oasis-open.org/ security/saml/v2.0/ saml-core-2.0-os.pdf.

[7] Netscape, SSL 3.0 Specification. http://www .netscape.com/ eng/ssl3.

[8] "Trusted Computing: Promise and risk". By Seth Schoen http://www.eff.org/Infrastructure/trusted_computing/20031001_tc.php

[9] Rick Kennell and Leah H. Jamieson. ***Establishing the genuinity of remote computer systems.*** In SSYM'03: Proceedings of the 12th conference on USENIX Security Symposium, pages 21–21, Berkeley, CA, USA, 2003. USENIX Association.

[10] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doom. ***Design and implementation of a tcg-based integrity measurement architecture.*** In SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium, pages 16–28, Berkeley, CA, USA, 2004. USENIX Association.

[11] Yoshihama Sachiko Ebringer T Maruyama Hirosh, Muneto Seiji. TPoD-trusted platform on demand. Joho Shori Gakkai Kenkyu Hokoku, pages 181–186, August 2004.

[12] Ahmad-Reza Sadeghai and Christian St¨uble. Property-based Attestation for Computing Platforms: Caring about Properties, not Mechanisms. In NSPW '04: Proceedings of the 2004 Workshop on New Security Paradigms, pages 67–77, New York, NY, USA, 2004. ACM Press.

[13] Elaine Shi, Adrian Perrig, and Leendert Van Doom. ***BIND: A Fine-Grained Attestation Service for Secure Distributed Systems.*** In SP'05: Proceedings of the

2005 IEEE Symposium on Security and Privacy, pages 154–168, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Xiao-Yong Li, Chang xiang Shen, and Xiao-Dong Zuo. An Efficient Attestation for Trustworthiness of Computing Platform. In IIH-MSP, pages 625–630,2006.

[15] Mimi Zohar, David Safford. *"A trusted linux client': IBM* Journal, 2005.

[16] Megumi Nakamura Sachiko Yoshihama, Tim Ebringer. ***Test and Analysis of Web Services.*** Springer Verlag, Springer Berlin Heidelberg, 2007.

[17] Zishuang (Eileen) Ye, Sean Smith, and Denise Anthony. ***Trusted paths for browsers.*** ACM Trans. Inf. Syst. Secur., 8(2):153–186, 2005.

[18] Trusted Network Connect. http://www.trustedcomputinggroup.com/tnc/

[19] IAIK: Institute for Applied Information Processing and Communications, Graz University of Technology. http://www.iaik.tugraz.st/.

[20] Jetty-Java-based Open Source Web Server. http:// www.mortbay.org/ jetty-61.

[21] Lobo - Java Web Browser. http://lobobrowser.org/java-browser.jsp

[22] Eclipse. Plug-in architecture for development. http://www.eclipse.org

[23] Trusted-Jetty. Trusted-aware Web Server. http://trusted-iettv.sourceforge.net/

[24] Trusted-Lobo. Open source Trusted Browser. http://trusted-lobo.sourceforge.net/

[25] Trusted Computing for the Java(tm) Platform. Available at,. http://trustedjava. sourceforge.net/.

[26] Secure Hash Algorithm 1. http://www.ietf.org/rfc/rfc3174.txt.

[27] Ryan Catherman David Safford Leendert van Doom David Challene, Kent Yoder. ***A Practical Guide to Trusted Computing.*** IBM Press, IBM T. J Watson Labs, USA, 2008.

[28] M. Alam, X. Zhang, M. Nauman, T. Ali, and J.P. Seifert. ***Model-based Behavioral Attestation.*** In Accepted for publication in SACMAT '08: Proceedings of the thirteenth ACM symposium on Access control models and technologies. Available at: http://serg.imsciences.edu.pk/pub/mba-sacmat08.pdf, New York, NY, USA, 2008. ACM Press.

[29] Shibboleth, Middle-ware architecture. http://mace.internet2.org/shibboletM

Appendix

Appendix A

Installations

Detailed installation of the prototype system

- TPM chip
- IMA
- Trusted-Jetty
- Trusted-Lobo

A.1 TPM

Most of the recent desktops come with a hardware chip embedded in it. This chip called TPM (Trusted Platform Module) and is activated from the BIOS setup of that system. For Linux operating system we need to install drivers in the kernel. TPM driver can be installed as kernel module or may be compiled into the kernel at compile time. For our implementation we need TPM to be installed into the kernel so that it is enabled at boot time and as soon as IMA take control it can concatenate the measurements into the TPM PCRs. For this purpose we need to recompile the kernel with the TPM driver enabled into the kernel.

Latest kernel is downloaded from the kernel.org and by making the kernel with

```
#make menuconfig
```

We nerd to enable the TPM into the drivers section of the kernel menu. As TPM is a char device so it resides in the char device section of the driver menu. After enabling the TPM driverjust recompile the kernel by doing

```
#make all; make modules_install; make install
```

After rebooting the system can be booted with the recompiled kernel version. Now the system is ready to be configured IMA installation.

A.2 IMA

The following instructions work for Fedora Core 6 but should be generic (you just need a configuration file that works for the kernel) latest patch of IMA for kernel version 2.6.24-3 is available at linux-ima.sourceforge.net.

I. Required kernel configuration options

a) crypto->SHA1 is (y)

[IMA needs sha before modules are loaded]

b) security->Default Linux Capabilities (n)

[IMA cannot share LSM with the capabilities]

c) choose (y) for "TCG run-time Integrity Measurement Architecture"

[switchtes IMA measurements on]

d) choose (y) for "IMA test mode"

This option tells IMA to try to use a real hw TPM or bypass it if in test mode. Choose (y) if you don't have a TPM on your machine or if you have a TPM on your machine but you want to test IMA and the dependencies first. In any case, make sure you have a TPM driver with the internal kernel interface patch posted to LKML 0512005. If you choose (n) and IMA can't start up for any reason, it panics the kernel to protect attestation (see below, 8.).

Say (n) only after testing your kernel configuration.

If unsure, say (y).

e) If you'd like to compile SELinux and IMA and choose between them at boot-time then configure:

NSA SELinux boot paramter

NSA SELinux boot parameter default value to (0) (see **3** for kernel command line options)

2. Compile and install the new kernel and initrd

```
make all; make modules_install; make install
```

3. Change kernel command line options

```
==================================
to activate the Integrity Measurement Architecture at boot-time,
add: 'ima=1', to deactivate use 'ima=0' (default)

If you have both SELinux and IBM IMA support compiled into the kernel, then switch at
least one of:
'ima=1 selinux=0' activates the Integrity Measurement Architecture
'ima=0 selinux=1' activates SELinux

You can't activate both because the kernel does not yet support LSM stacking.
4. Trouble-shooting (restart your system to activate new kernel)
==========================================================
Use 'dmesg |grep IMA` to print IMA status startup information:
You may find the following output:
  a) you are fine if you see the following lines (if you have TPM hardware):
  ----
  IBM Integrity Measurement Architecture (IBM IMA vx.x mm/dd/yyyy).
     IMA (test mode)
  ----
  or the following lines (if you don't have TPM hardware):
  ----
  IBM Integrity Measurement Architecture (IBM IMA vx.x mm/dd/yyyy).
     IMA (test mode)
     IMA (TPMIBYPASS - no TPM chip found)
  ----

  b) you need to add the "ima=1" kernel boot paramter if you see:
  ---
  IBM Integrity Measurement Architecture (IBM IMA vx.x mm/dd/yyyy).
     IMA (not enabled in kernel command line) aborting!
  ---
```

c) you need to disable security->Default Linux Capabilities or
SELinux (see configuration requirements) if you see:

```
---
IBM Integrity Measurement Architecture (IBM IMA vx.x mm/dd/yyyy).
  IMA (test mode)
  IMA (LSM/not free) aborting!
---
```

A.3 Trusted Jetty

....
Install lnstmctions for Trusted Jetty

Trusted-jetty is a branch of the popular java-based browser 'jetty' which is modified to support remote attestation (Remote attestation is a part of 'Trusted Computing (TC) which allows a remote party to verify that a trusted environment exists on a target platform. See [4] for details regarding this novel concept.)

Trusted-jetty relies on the BAClientDC package (which can be downloaded from the trusted-jetty sourceforge project site.) Following are the details regarding the requirements for installing and using Trusted Jetty.

Requirements:

1. Java SE 5+ SDK (We have tested trusted jetty with Sun Java SE 5. Use another VM at your own risk.)

2. BAClientDC package (available at http://trusted-jetty.sourceforge.net)

3. Jetty server version 0.6.0 with all dependencies

* A client machine with trusted-lobo (a java-based remote attestation-enabled browser available at http://trusted-lobo.sourceforge.net)

Installation Instructions:

I. Install Java SDK

2. Patch the source of jetty server with the patch provided in this archive. In linux, this command would look like:

```
$JETTY_EXTRACTED_DIWjetty0.6 $> patch -pl --dry-run < ../trusted-jetty-0.6.patch
```

to check if the patch works. Then, to apply the patch:

```
$JETTY_EXTRACTED_DIR/jetty0.6 $> patch -pl < ../trusted-jetty-0.6.patch
```

3. If you're running eclipse, add the BAClientDC package in the 'project dependencies'. If you do not want to use eclipse, add the BAClientDC class files in the java classpath.

4. Start the jetty server on any port. Wait for a trusted browser to connect and see what happens.

If you have any queries, visit our site at http:llserg.imsciences.edu.pk or join the trusted-jetty mailing list and post your query there.

A.4 Trusted-Lobo

Install Instructions for Trusted Lobo

Trusted-lobo is a branch of the java-based browser 'lobo' which is modified to support remote attestation (Remote attestation is a part of Trusted Computing (TC) which allows a remote party to verify that a trusted environment exists on a target platform. See [1] for details regarding this novel concept.)

Trusted-lobo relies on the BAClient and BAClientD packages (which can be downloaded from the trusted-

lobo sourceforge project site.) Following are the details regarding the requirements for installing and using Trusted Lobo.

Requirements:

1. Java SE 5+ SDK (We have tested trusted jetty with Sun Java SE 5. Use another VM at your own risk.)

2. BAClient and BAClientD packages (available at http://trusted-jetty.sourceforge.net)

3. Lobo browser version 0.6.0 with all dependencies

* A server machine with trusted-jetty (a java-based remote attestation-enabled web server available at http://trusted-jetty.sourceforge.net)

Installation Instructions:

1. Install Java SDK

2. Patch the source of jetty server with the patch provided in this archive. In linux, this command would look like:

```
$LOBO_EXTRACTED_DIR/lobo0.6 $> patch -p1 --dry-run < ../trusted-lobo-0.6.patch
```

to check if the patch works. Then, to apply the patch:

```
$LOBO_EXTRACTED_DIR/lobo0.6 $> patch -p1 < ../trusted-lobo-0.6.patch
```

3. If you're running eclipse, add the BAClient and BAclientD packages in the 'project dependencies'. If you do not want to use eclipse, add the class files of the two packages in the java classpath.

4. Start the jetty server on any port. Connect to the server using lobo and wait for the response.

If you have any queries, visit our site at http://serg.imsciences.edu.pk or join the trusted-jetty mailing list and post your query there.