Practical Property-based Attestation on Free Software

Liang Gu[1], Anbang Ruan[1,2], Yao Guo[1], Qingni Shen[1,2], Xiangqun Chen[1]
[1]Key Laboratory of High Confidence Software Technologies (Ministry of Education),
Institute of Software, School of EECS, Peking University.
[2]School of Software and Microelectronics, Peking University.
Beijing. China.
[1]{guliang05,yaoguo,cherry}@sei.pku.edu.cn, [2]{ruanab, shenqn}@infosec.pku.edu.cn

***Abstract*—Existing attestation schemes still tie the property with binary code of program and they have to hold a large number of known-good measurements. However, free software is usually configured and optimized by end user and it is impossible for existing schemes to attest the customized free software of unpredictable versions. We introduce a source code based attestation scheme on free software. The binary code of a program is bound with its source code and building configuration. The property of a program is obtained by examining its source code and building configuration, and a certificate is issued by a trusted verifier. We implement our prototype based on Gentoo and its Portage, which is a source code based package management system. Our scheme makes an effort to provide a key TCG feature—-remote attestation for free software community.**

***Keywords*-remote attestation; trusted computing; property based attestation; free software;**

I. Introduction

Many IT systems nowadays are conducted on open computer platforms across heterogeneous domains or over the public Internet. Entities in such open, distributed and dynamic environment usually behave on their own behalf and may not trust each other for mission-critical operations or transactions. Remote attestation provides an important way to establish trust on parties in open network. In TCG's trusted computing standard, remote attestation allows a challenging platform, usually referred to as a *challenger*, to verify the configuration integrity of a remote platform (i.e. an *attester*). Recent years have witnessed various evolutions out of the basic TCG attestation in many dimensions, including program semantics attestation [5], security policy enforcement [7], property attestation [3, 8].

It is still difficult for these existing schemes to handle some problems to achieve practical attestation solutions for programs, especially for free software. First, as most of existing schemes are based on binary code of software, it is difficult to hold a known-good measurement database for so many different programs with various versions. Although existing property-based attestation schemes [3, 8] introduced the concept of attesting programs based on their properties, these stated properties are still tied to the binary codes of programs. As a result, these property-based attestation still need a giant known-good binary database, which is required by the Trusted Third Party for binary code certification. Second, it is difficult to conclude with certain properties by simply checking the binary code of the specified software. Third, in some cases, it is even impossible to hold all known-good measurements for possible binaries. Some programs are distributed via source code, like free software. End users may configure and tailor the source code to build the binary code as a customized version and it is impossible to hold binary codes of all possible versions. So it is difficult for another party to attest integrity of customized free software by only employing existing attestation solutions. As a result, existing attestation schemes are not adequate for free software.

We propose a source code based attestation to solve above problems and initiate an effort for applying the key TCG feature—-attestation in free software community. Our scheme binds the binary code with its corresponding source code and building configurations, and the properties of specified binary code can be certified by a trusted verifier via checking the source code and building configuration. At runtime, the challenger only has to verify the property certificate of target binary code. Our scheme applies for free software and also support practical property certificate for ordinary programs.

II. Scheme Design

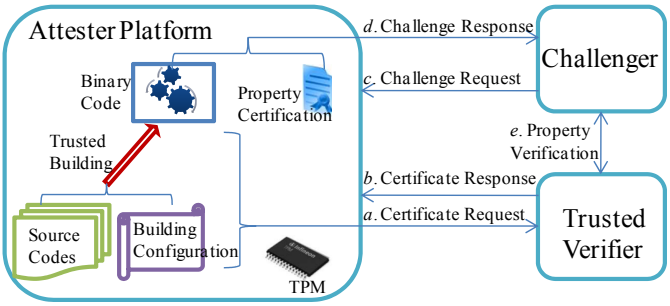

Fig. 1. Source code based attestation framework

A. Attestation Framework

Our source code based attestation framework is shown in Figure 1. Three parties are involved in our scheme: The challenger, the attester and the trusted verifier. A typical scenario is as follows: the attester configures and builds a free software P on its platform; at runtime, the challenger wants to attest the property of this customized program; the challenger and attester will carry out the attestation procedure for this free software with the help of the trusted verifier.

B. Attestation Procedure

For a program P, its binary code BC is built from corresponding source code $SCs = \{sc_1, sc_2, ..., sc_i\}$ with specified building configuration CF, where sc_i denotes a source code file in all P's source code. In our scheme, we bind the binary code with its source code and building configuration, and we bind the property of software with its source code and building configuration. Two phases are involved in our scheme:

certification phase (step *a*, step *b* in Figure 1) and attestation phase (step *c*, step *d*, step *e* in Figure 1).

In the certification phase, the customized P is certificated by a trusted verifier. First, the attester platform carries out a trusted building process to generate the binary code from the customized source code and building configuration. At the end of the trusted building process, P's BC is bound with SCs and CF with a TPM signature as follows :

$$HBC = SHA1(BC);$$
$$HSCs = SHA1(sc_1||sc_2||...||sc_i);$$
$$HCF = SHA1(CF);$$
$$TPM\_extend(HBC||HSCs||HCF);$$
$$SIG_P = sig_{AIK_{priv}}(HBC||HSCs||HCF).$$

Then, attester sends certificate request to a trusted verifier (step *a*) with following messages: $\{BC, SCs, CF, SIG_P, SML\}$, where SML is for Stored Measurement Log. Usually it is not necessary to send all SCs. Only an update based on a standard version is required, like the Linux kernel patch. The trusted verifier can conclude with certain properties for P by examining the received messages. With a successful result, trusted verifier issues a certificate on P and returns it to the attester platform (step *b*):

$$cetr_{PK}(TV, pp, HBC) = (HBC, HSCs, HCF, pp, SIG_P)$$

where PK is the public key of trusted verifier TV, pp is a property, $cetr_{PK}(TV, pp, HBC)$ is a property certificate for P.When P with certain SCs and CF is found to be vulnerable, trusted verifier has to follow a revocation protocol to revoke corresponding property certificate.

In the attestation phase, when the challenger requests for an attestation on the program(step *c*), the attester platform may response with the program's integrity measurement and certificate(step *d*). Then the challenger may verify the certification from the trusted verifier(step *e*).

C. Attester Platform

The attester in our model has full privileges in controlling the software system on its platform (Figure 2), except the Secure Virtual Machine, TCG software and Attestation Agent. With TPM and TXT[6]/SVM[1], the attester platform may provide both static root of trust and dynamic root of trust, and we may establish the trust on the integrity of a trusted domain based on Secure Virtual Machine.

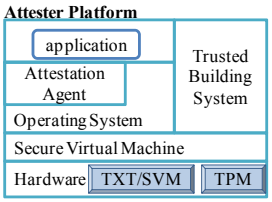

Fig. 2. Architecture of attester platform

Two separated domains are supported by the Secure Virtual Machine on attester platform. One is a normal domain for ordinary operating system. We introduce an attestation agent as kernel module in the OS for runtime measurements on applications. With support of TPM and Secure Virtual Machine, the integrity of attestation agent can be recorded for attestation each time it works.

Another is a secure domain which hosts a Trusted Building System (TBS), and this TBS provides a trustworthy process for building these customized source code into binary code. The Secure Virtual Machine leverages the TXT/SVM facilities to provide a trusted domain for TBS. The building process can be attested to prove its trustworthy.

D. Property Certification

For program with only variant building configurations, trusted verifier can employ automatic tools to examine the building configurations. For program with source code modifications at lower granularity, like instruction, trusted verifier requires more sophisticated verification techniques for program certification, like program transformation.

III. Implementation

We employ XEN[2] supported by TXT facility as the Secure Virtual Machine. Gentoo Linux [4] is chosen as the operating system to host our prototype. Gentoo and its applications are distributed as source code and can be optimized and customized for just about any application. In our implementation, Gentoo is used in two ways: one for ordinary applications in the ordinary domain of XEN and another for the TBS in a protected domain. For the ordinary one, we introduce the attestation agent module as a Linux Security Module to monitor and record the execution of applications. For the TBS, we configure the Gentoo to have minimal kernel to support its Portage package management system and the Portage plays the key role in TBS to build source code. We introduce a module in the reduced Gentoo to employ TPM facilities for monitoring and recording the building process. We will show our implementation result later in a full version.

IV. Conclusion and Future Work

With the support of our scheme, it is possible for free software community to employ the key TCG feature—-remote attestation, to support trust establishment on applications in open networks. After we finish the implementation, we will make it available for the free software community.

V. Acknowledgements

This work is supported by the National Basic Research Program of China (973) under Grant No. 2009CB320703 and the Science Fund for Creative Research Groups of China under Grant No. 60821003.

References

[1] AMD. AMD64 Virtualization Codenamed "Pacifica" Technology–Secure Virtual Machine Architecture Reference Manual. Technical Report Publication Number 33047, Revision 3.01, AMD, May 2005.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP*, volume 37, 5 of *Operating Systems Review*, pages 164–177, New York, October 19–22 2003. ACM Press.

[3] Liqun Chen, Rainer Landfermann, Hans Löhr, Markus Rohe, Ahmad-Reza Sadeghi, and Christian Stüble. A protocol for property-based attestation. In *STC '06*, pages 7–16, New York, NY, USA, 2006. ACM Press.

[4] Gentoo. Gentoo Linux. http://www.gentoo.org/, 2009.

[5] Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic remote attestation—a virtual machine directed approach to trusted computing. In *the Third virtual Machine Research and Technology Symposium (VM '04). USENIX.*, 2004.

[6] Intel Corporation. Intel trusted execution technology — preliminary architecture specification. Technical Report Document Number: 31516803, Intel Corporation, 2006.

[7] Trent Jaeger, Reiner Sailer, and Umesh Shankar. PRIMA: policy-reduced integrity measurement architecture. In *SACMAT '06*, pages 19–28, New York, NY, USA, 2006. ACM Press.

[8] Ahmad-Reza Sadeghi and Christian Stble. Property-based attestation for computing platforms: caring about properties, not mechanisms. *New security paradigms*, 2004.