Realizing Macro Based Technique for Behavioral Attestation on Remote Platform

Chapter · January 2021

DOI: 10.1007/978-3-030-55180-3_10

5 authors, including:

Alhuseen Omar Alsayed
King Abdulaziz University
26 PUBLICATIONS **139** CITATIONS
SEE PROFILE

Muhammad Binsawad
King Abdulaziz University
16 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Jawad Ali
University of Kuala Lumpur
18 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Ahmad Shahrafidz Khalid
Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique, d'H…
6 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Realizing Macro Based Technique for Behavioral Attestation on Remote Platform

Alhuseen Omar Alsayed[1], Muhammad Binsawad[2], Jawad Ali[3]*(✉), Ahmad Shahrafidz Khalid[3], and Waqas Ahmed[4]

[1] Deanship of Scientific Research,
King Abdulaziz University, Jeddah 21589, Saudi Arabia.
aoalsayd@kau.edu.sa,
[2] Faculty of Computer Information Systems,
King Abdulaziz University, Jeddah 21589, Saudi Arabia
mbinsawad@kau.edu.sa,
[3] Malaysian Institute of Information Technology,
Universiti Kuala Lumpur, Malaysia.
jawad.ali@s.unikl.edu.my; ahmads@unikl.edu.my,
[4] UniKL Business School, Universiti Kuala Lumpur, Malaysia.
waqas.ahmed@s.unikl.edu.my,

Abstract. In Trusted Computing the client platform is checked for its trustworthiness using Remote Attestation. Integrity Measurement Architecture (IMA) is a well-known technique of TCG based attestation. However, due to static nature of IMA, it cannot be aware of the runtime behavior of applications which leads to integrity problems. To overcome this problem several dynamic behavior-based attestation techniques have been proposed that can measure the run-time behavior of applications by capturing all system-calls produced by them. In this paper, we have proposed a system call based technique of intrusion detection for remote attestation in which macros are used for reporting. Macros are used to denote subsequences of system calls of variable length. The basic goal of this paper is to shorten the number of system calls by the concept of macros which ultimately reduces the processing time as well as network overhead.

Keywords: Remote Attestation, Dynamic Behavior, Intrusion Detection, Trusted Computing

1 Introduction

Nowadays the world is executed by computing technologies, security is exceedingly important. Existing IT and computing infrastructure become more intricate than the past. Many technologies have been developed, such as cloud computing, software as a service(SaaS), cloud formation, e-commerce, and virtualization, etc. These technologies facilitate the end users as well as the IT

*Corresponding author.

staff like server maintainer and software developers. On the other side, the enhancement in these complex software stacks results in open doorways for new vulnerabilities. The question arises here is to ensure that the remote machine while communicating with, is trusted or not?

Trusted Computing is a known concept in today's computing infrastructure which helps in integrating hardware-based security in the existing security framework [1]. The most important feature of TCG technology is to embed hardware root of trust inside the computing platforms. For this purpose, TCG introduces a cryptographic co-processor chip called *Trusted Platform Module* (TPM) [2]. There are a number of tamper-proof locations called *Platform Configuration Registers* (PCRs) inside TPM. These PCRs can store platform configuration in the form of cryptographic hashes. SHA-1 is used as a one-way hash function that cannot be removed or changed by any software application. These stored hashes inside PCR can further be reported to a remote system in a secure channel. The process by which these hashes are reported to a remote platform in order to build trust is called *remote attestation.*

Remote Attestation is one of the known features of Trusted Computing which can be used to verify the integrity of remote systems, in order to build trust between them. There are several kinds of remote attestation techniques i.e. Static remote attestation [3] and dynamic behavior-based remote attestation [4–7]. A well-known technique for remote attestation is *Integrity Measurement Architecture*(IMA). Reiner Sailer et al [3] designed an integrity measurement system and implemented it into Linux. IMA considers to be the first technique for remote attestation. In IMA the hashes of every executable loaded for execution are calculated and stored to a log file called SML (Stored Measurement Log). When the system boots, a chain of trust is formed by first computing the hash (SHA-1) of BIOS stored in TPM. The BIOS then passes the control to Boot Loader. The Boot loader measures and calculates hash of the Operating System kernel and stores the hash in PCR-10. The kernel is then responsible for loading further executables such as init and libraries. The kernel first measures the hash of every executable and then allows the Operating System for loading. A log of these hashes gets maintained in a log file called stored measurement Log (SML). All these hashes are then aggregated and stored in TPM using PCR. After each calculated hash a PCR-extend function is called for concatenating the previous hash with the current hash to form a single hash. In order to verify the system remotely, the challenger first sends the attestation request. In response attesting system sends an attestation token which includes PCR-10 value and SML to challenging party. The challenger calculates all values of SML in the same order and compares with PCR-10, so if both the values are same then the remote party is considered to be trusted. But due to the static nature of IMA there are some limitations. It can only measure the load time measurements and cannot be well informed of runtime behavior. Another weakness of IMA is, it cannot have resistance to buffer overflow attacks and return-oriented programming [8][9]. Therefore, there is a need for dynamic behavior attestation mechanism to handle the said issues.

The most well-known technique that is used to measure the dynamic or runtime behavior of an application is through sequences of system calls generated by an application during its lifetime [10] [9]. These techniques are effectively working on host-based security as intrusion detection systems (IDS), but there is no implementation in remote attestation scenarios where it is capable to report the behavior measurement to a remote party for verification. However, every single application generates a huge number of system calls in a small time-stamp [11]. This results in more processing time on the target platform that ultimately leads to a network overhead during transmission.

Outline: The rest of the paper is organized as follows: In Section 2 a description regarding our contribution to existing techniques of remote attestation is mentioned. Section 3 discussed the background and some of the literature studies about remote attestation techniques along with its limitations. In the same section we also provided some study of the intrusion detection systems to understand our proposed architecture. Further in Section 4 we define behavior in terms of macros and Section 5 elaborate Linux Security Module structure. Section 6 describes our proposed architecture of remote attestation along with analysis with the existing techniques. Finally Section 7 concludes the paper.

2 Contribution

In this research, we have proposed an existing technique of intrusion detection presented by [12] in a remote attestation framework in order to address the above problems. Afterward, we investigate how to reduce system call log by using the concept of macros. The log reduction will result in below:

- Increasing the efficiency of behavior measurement on remote end.
- Reducing log being sent to a challenger for verification would decrease the network overhead.

3 Background

3.1 Trusted Platform Module

Trusted Platform Module (TPM) has been introduced by TCG [13] specifications as a basic component. TPM is a cryptographic co-processor chip that secures the data and provides hardware root-of-trust. Almost all vendors laptops and desktops have TPM inside of them. TPM has several components inside that are RSA Engine, Random Number Generator (RNG), SHA-1 & HMAC Engine, CPU, Volatile & Non-volatile memory RSA is a hardware engine being used in public-key cryptography and is one of the essential requirements for TPM. It is used to generate new keys for signing purposes as well as encryption and decryption of other keys. Additionally, TPM can support different key size that is 512bits, 1024bits, and 2048bits. The SHA-1 algorithm is used to compute 160-bit hashes of data. RNG is used to create keys as well as nonces (numbers used once) during attestation. It can also be applied through the use of software-based RNG which gets sourced from hardware RNG. HMAC is used as a keyed hash function that incorporates a cryptographic key for converting unkeyed hash function to keyed hash function. It is used for both to verify data integrity as well as authentication of a user. Authdata is a 160bit secret code produced from the new-key which is generated inside of TPM. All these above functions are related to cryptographic capabilities of TPM.

There are a number of shielded locations inside TPM called *platform configuration registers* (PCRs). It plays a very important role in completing the process of remote attestation securely and reliably. PCRs are used to store measurements in the form of hashes. There can be a total number of 16 or 24 PCRs depending on the specification that is used to design TPM. PCRs 0 to 7 are reserved for pre-execution which shall build a chain of trust before the OS gets control. Several functions are used in TPM based operations. One of the important functions used while remote attestation is *PCR_Extend* that can append and aggregate the hashes in TPM.

Protection through hardware means that the information from the user will be stored accurately and signed by the TPM Key-pair normally called storage root key (SRK). This particular key is attached to TPM and could not get back from TPM. SRK can be used either directly to encrypt the data or for securing other keys called storage keys. SRK can be washed from BIOS through a TPM specific instruction i.e. TPM-ForceClear. The public endorsement key is used for binding data and can only be unbinded by private pair of public key. Similarly, sealing is done through public key as in binding but the unsealing operation is different from binding. A hint or some special instruction is given for unsealing e.g. nonce

3.2 Dynamic Behavior Attestation

Integrity Measurement Architecture (IMA) [3] is a well-known technique of static remote attestation for verifying target platform by reporting the hashes of applications. Hashes of applications help out IMA to verify the trusted state of the target system upon the request of challenger. Various issues came out due to this hash-based technique, one of them is highly rigid target domains. As a solution to these problems variety of attestation techniques have been proposed. Such techniques are based upon the runtime attitude of applications, data structures, and system call sequences.

Jeager et al. [8] proposed an extension to the IMA called Policy Reduced Integrity Measurement Architecture (PRIMA). It tried to overcome the issues of IMA that is:

- It computes load time measurements of code which does not accurately reveal the runtime behaviors of an application.
- There is no support to verify the integrity of some specific application rather than the whole system that needs to be verified.

- IMA cannot reduce the list of measurement by measuring those applications which are associated with the target application.

PRIMA measures the integrity of target applications by information flow. It does not measure only the code but also has awareness of information flow between processes. Its prototype is implemented through SElinux policy for providing the information flow that would produce some nature of dynamism in attestation mechanisms. The additional requirements for information flow are load time MAC policy and trusted subjects, mapping between MAC policy and code. The author proved that it can attest CW-Lite (short version of Clark-Wilson). Moreover, PRIMA approach addresses some issues found in previous techniques such as false negative attestation, false positive attestation and also decrease the number of necessary measurements. However, there are some drawbacks in this technique as given:

- PRIMA is still dependent on hashes of code, policy, and files.
- It cannot capture the dynamic behavior of an application.

Liang Gu et al [9] proposed remote attestation on program execution which is a step towards dynamic behavior-based attestation. They capture the behavior of the application by collecting system calls with hardware root-of-trust. Although there are several solutions for monitoring behavior of application which provide security for the system, but the software-based security is not so feasible than hardware-based solution. However, there are some limitations to this approach.

- enormous number of system calls can cause performance overhead on the target to be measured.
- Reporting of this immense number of system calls results in network overhead.
- Solely system calls cannot give any meaning in verifications, unless there is some sort of patterns i.e. sequence of system calls that capture the malicious behavior.

Loscoco et al [14] developed a framework called Linux Kernel Integrity Measurement (LKIM) which is considered as a step towards behavioral attestation. It examines some critical running data structures and plots them on a graph for decision making. LKIM is based on contextual inspection that is used for measuring the components of running kernel. It can execute in both environments: as a user-process in base environment and domain in the hypervisor environment. LKIM monitors the running kernel so that it measures the components in the current state. Although this is quite a better approach but some limitations are still attached to it.

- As it measures the hashes of running kernel and produces a big amount of logs, in a result the network traffic increases and becomes a bottleneck over the network.

– LKIM analyzes kernel data structure at runtime that needs an extra amount of time for processing on the target end. However, due to the static nature of IMA it cannot be aware of the runtime behavior of applications which leads to integrity problems. To overcome this problem several dynamic behavior-based attestation techniques have been proposed which can measure run-time behavior of applications by capturing all system-calls produced by them [10] [9] [15].

3.3 Intrusion Detection System

To monitor activities of a network or a computer system and to analyze them as whether the system or network is acting in malicious or normal way is known as Intrusion Detection System (IDS). Depending on classification there are two major types of intrusion detection system i.e. Host Based IDS & Network Based IDS [16]. Host based intrusion detection system (HIDS) provides protection and security against an individual host. Network intrusion detection system (NIDS) is used to monitor and protect traffic from the entire network and generate alarms or response against malicious attempt.

Based upon the detection techniques of intrusion detection system, there are two main categories of IDS i.e. Misuse IDS & Anomaly based IDS. Misuse IDS stores the known signature of intrusions. Whenever an action occurs and it matches with the previously experienced signature, is considered as misuse detection [17]. It reports the event which is matched as intrusion. Although this approach is consider to be better for accuracy with low false positive, but it cannot take action (detect) on new or unmatched pattern. In addition, while adding new signatures regularly will lead to performance overhead on the underlying system. To overcome this issue, anomaly based detection system is based on the behavior of system which creates normal behavior profiles for the system. When an event occurs in the system, it observes and gets compared with the normal behavior profile and report the [17] unusual deviations as intrusion. User behavior can be created or profiled through statistical methods, inductive pattern generation, data mining, machine learning and neural networks.

Kosoresow et el. [12] shows his preliminary work for analyzing system calls sequences for normal and abnormal behavior. They figured out *macros* which are generated by taking common prefixes, suffixes and repeating strings in a system call trace. Each macro consists of variable length pre-defined patterns of system calls and every application has its own set of macros.

Forrest et al. [18] introduces the idea of using short sequences of system calls of runtime (privileged) processes, which results in generating a stable signature for normal profile. Every program in execution produced a sequence of system calls and determined by their order in which they are executed. For any significant program, there will be a trace of system calls which have not been observed. However, short-range of system call sequences are notably consistent and results in defining normal behavior. To create a separate profile of normal behavior for each process, authors define system calls in the form of fixed length short sequences i.e. 5, 6 and 11.

4 Defining Macros as a Behavior

When an application runs, it generates system calls for performing different tasks. These system calls are matched against pre-defined macros to determine the sequence of macros. Macros are the regular pattern occurring frequently in a system call traces. In this case, its size is variable and each macro is denoted by an alphabet i.e. A, B, C, D to Z [12]. In this proposed research the traces of system calls are matched against the pre-defined macros which result in producing macros. For example, if an application generates a trace, it will be checked in the list of macros and finally generate a sequence of macros. Figure 1 illustrates the mechanism of producing macros from a system-calls sequence.

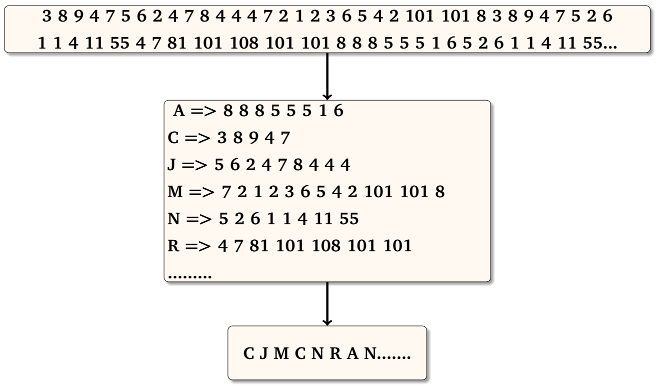


Fig. 1: System Calls to Macros transformation

5 Linux Security Module (LSM)

The most important feature provided by Linux kernel is the Linux Security Module(LSM). By default Linux is set up with *Discretionary Access control* (DAC) system. DAC is the first model towards access control mechanism and adopted by the Linux OS. DAC is an owner-centric security model which means that it restricts or grants access to an object by the policy defined by owner of the object [19]. For example, a user A creates a file then A will decide that what kind of access to file is provided to other users i.e. groups or others. A root user also called as privileged user has all the access rights on system. In case, if malicious user got login as root so there is no restriction in DAC to prevent it.

The LSM is a general framework based on Mandatory access control (MAC) which provides a base to third party security modules for implementing their own defined policies for carrying out any action in the system. The actions are determined by designers of the framework which required authorization. These actions are first passed through the LSM framework before the kernel finished the task on behalf of an application. Since, in our proposed work we are trying to intercept calls with the help of a custom LSM module. This Module first creates macros from system calls trace and stores in a log file called *security-fs*. Before storing these macros we will take measurement in the form of hashes and store them to PCR. This process will be carried out through the attestation module in our proposed framework. Every application has its own fixed possible

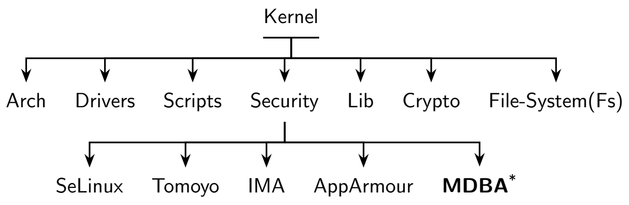


Fig. 2: Linux Kernel Directory Structure

number of system calls that are generated during execution. We define macros of application on the target platform in a database. With the help of this macro database, we will identify the known and unknown macros generated from the trace.

6 Proposed Architecture

The proposed architecture as shown in Figure 3 consists of two entities i.e. target & Challenger. The first and focused entity in architecture is the target or client platform. The measurement process of application behavior is performed on a target platform which will further be used for verification on challenger-side. The second entity is the challenger who wants to know the trustworthiness of the client system. On target-end, a new custom LSM module named *Macro based Dynamic Behavior Attestation* (MDBA) is proposed as shown in figure 2, which will directly be connected with LSM hooks for creating macros rather than performing mapping between system-calls and LSM hooks. In contrast to earlier remote attestation mechanisms where every system call is considered to be measured, this proposed technique first creates macros that are pre-defined in the database and calculates the measurements (Hashes) of these macros. Further, these measurements will be stored in SML and their aggregated hash value will be stored in PCR.

6.1 Reporting Macros for Verification

Generally, a remote attestation mechanism establishes in request and response manner. To perform the attestation process, the first challenger will send an at-

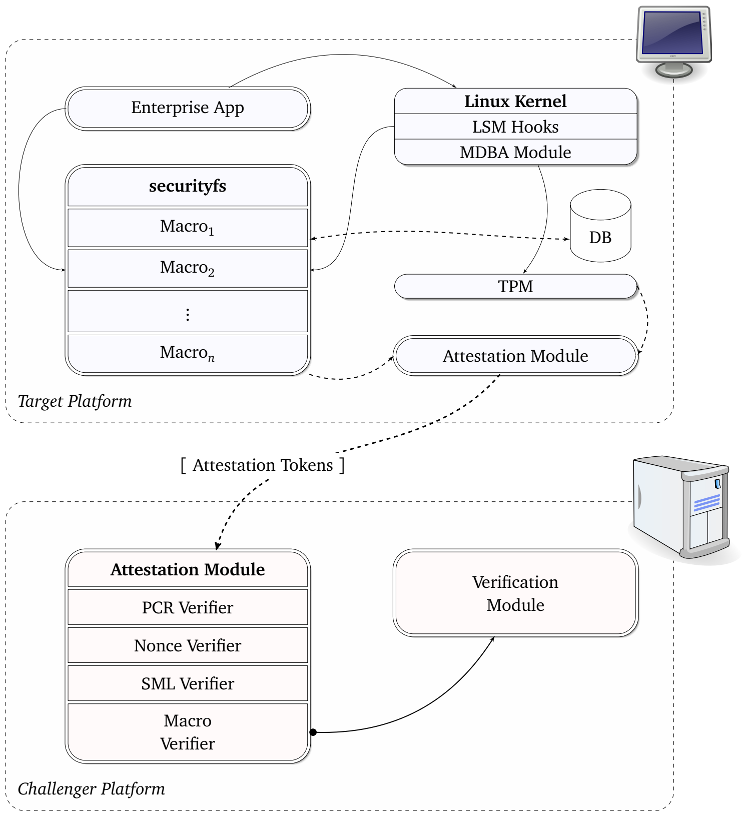
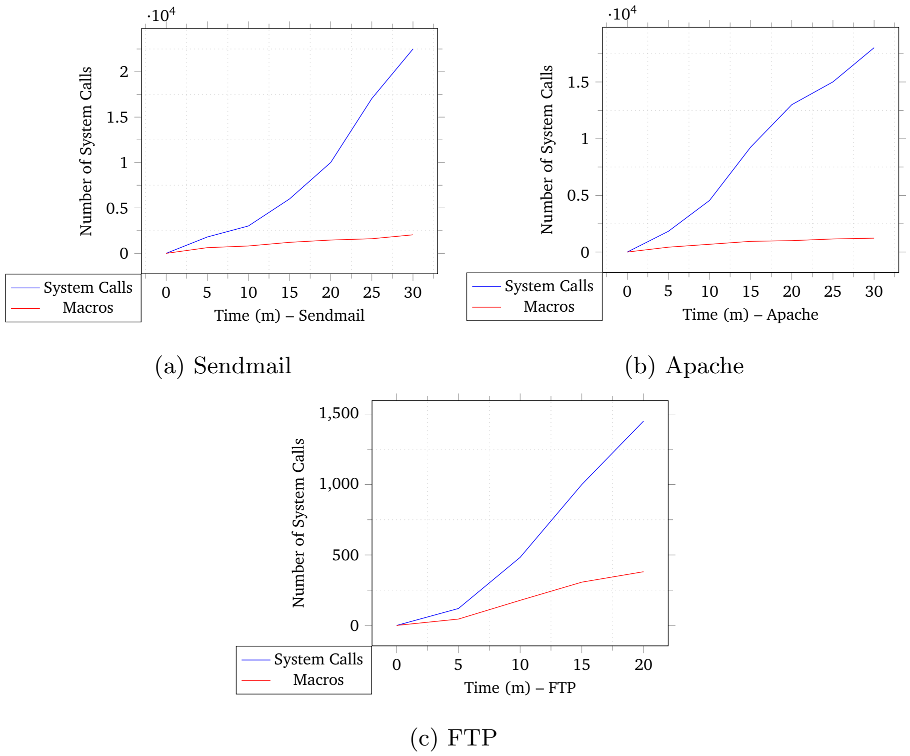


Fig. 3: Proposed Architecture

testation request along with nonce to the target system for verifying trustworthiness. The target system receives an attestation request and prepares a response that contains all measurement logs along with the PCR-12 Quote. The TPM prepares this response at the target end. TPM first calculates the PCRComposite structure over specifics PCRs. There are two PCRComposite structures calculated by attestation module: one is PCR-10 which is used to store system static measurement, while the other PCR-12 is used to store the measurement of macros of the client application. Afterward, TPM calculates hashes for each structure and appends these hashes to a fixed value along with the nonce received from the challenger side in attestation request. TPM then sign the values of PCR Composite using the private part of the Attestation Identity Key (AIK). Finally, the challenge-response is made and sent to the challenger that contains PCR Composite structures with their digital signatures. Now when the response is received, the challenger first recomputes hashes from the log (SML) and compares them with the PCR Quote, If both the values match to each other then the target will be considered trustworthy. This process will take place through the attestation module on the challenger end.

6.2 Comparisons of Results with Existing Techniques

Generally, every application generates a very large number of system calls in a short time. As a result, the system call log will also increase to an unbounded size. Reducing the log by introducing our proposed technique is one of the main objectives of this research. As discussed earlier, our technique maps the system calls traces of an application against the *macros* (variable length system call sub-sequences).



(a) Sendmail (b) Apache

(c) FTP

Fig. 4: System Call VS Macros Comparison

Different applications have been analyzed to show the improvement over the existing system calls based techniques as shown in Figure 4. Initially, a *send-mail* application is selected for testing as a target application (cf. Figure 4a). After running the send-mail applications for 5 minutes, it generates almost 1800 system calls while their corresponding macros were 625 which is three times less than the actual number of system calls. This improvement will reduce the measurement size as well as a performance by lesser SHA-1 and PCR-Extend operations in TPM. After executing the application for a further 15 minutes, it generates about 6000 system calls while 1200 macros were seen.

And after half an hour the application produces 22500 system calls and the final value for macros was 2100. Further, *FTP-Client* application has been considered for testing. In the first five minutes of FTP session, it produces 120 system calls while 45 macros were found. After watching for 15 minutes it generates 1000 system calls while the number of macros was 300 as shown in Figure 4c. Finally, *apache* application is monitored for about 30 minutes. Apache generates almost 18000 system calls while their corresponding macros were found almost 1250 as illustrated in Figure 4b. The overall results from these applications show a significant decrease in the number of measurements. Furthermore, Table 1 shows the significance of our proposed approach in terms of minimizing the log. The reduction in system calls will also decrease the reporting log by sending only macros rather than the whole system calls log.

Table 1: System Calls and Macros Log Size Comparisons

Times (min)	**Target Application**	**System Call Log Size**	**Macros Log Size**
5	FTP	0.31KB	0.23KB
15	FTP	14.34KB	10.90KB
25	FTP	24.77KB	11.43KB
5	SendMail	5.75KB	4.20KB
15	SendMail	19.50KB	14.78KB
25	SendMail	36.23KB	21.63KB
5	Apache	3.61KB	0.41KB
15	Apache	20.10KB	9.63KB
25	Apache	38.47KB	20.03KB

7 Conclusion

In this paper, we have presented a new technique of remote attestation which utilizes an existing intrusion detection system to measure the dynamic behavior of the remote application. We have discussed the workflow of the model along with the implementation plan in detail. We have studied the existing dynamic behavior attestation techniques and figure out their limitations which are the main bottleneck of these techniques to be implemented in the real scenarios. Using this simple technique proposed we can record and measure the dynamic nature of the remote platform. The most prominent aspect of our proposed solution is that system calls traces of an application are matched against a variable-length pattern of system calls called *macros*. Afterward, measurements of the macros are extended in TPM and their log is maintained in the SML-store measurement log. Representation of system calls by using macros reduced measurement and their log file sizes to an optimal size.

Our future work includes the real-time applications of the proposed architecture in various use-cases. We will give a detailed analysis of different other applications and show the usability of this work. We will provide an open-source implementation for getting feedback from the research community.

References

1. C. Mitchell, C. Mitchell, and C. Mitchell, "Trusted computing." Springer, 2005.
2. S. Bajikar, "Trusted platform module (tpm) based security on notebook pcs-white paper," *Mobile Platforms Group, Intel Corporation (June 20, 2002)*, 2002.
3. R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a tcg-based integrity measurement architecture."
4. T. Ali, J. Ali, T. Ali, M. Nauman, and S. Musa, "Efficient, scalable and privacy preserving application attestation in a multi stakeholder scenario," in *International Conference on Computational Science and Its Applications*. Springer, 2016, pp. 407–421.
5. T. A. Syed, S. Jan, S. Musa, and J. Ali, "Providing efficient, scalable and privacy preserved verification mechanism in remote attestation," in *2016 International Conference on Information and Communication Technology (ICICTM)*. IEEE, 2016, pp. 236–245.
6. T. Ali, M. Zuhairi, J. Ali, S. Musa, and M. Nauman, "A complete behavioral measurement and reporting: Optimized for mobile devices," *COMPSE 2016 - 1st EAI International Conference on Computer Science and Engineering*, 2017.
7. T. A. Syed, R. Ismail, S. Musa, M. Nauman, and S. Khan, "A sense of others: behavioral attestation of unix processes on remote platforms," in *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '12. New York, NY, USA: ACM, 2012, pp. 51:1–51:7. [Online]. Available: http://doi.acm.org/10.1145/2184751.2184814
8. T. Jaeger, R. Sailer, and U. Shankar, "Prima: policy-reduced integrity measurement architecture," in *Proceedings of the eleventh ACM symposium on Access control models and technologies*. ACM, 2006, pp. 19–28.
9. L. Gu, X. Ding, R. H. Deng, B. Xie, and H. Mei, "Remote attestation on program execution," in *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, ser. STC '08. New York, NY, USA: ACM, 2008, pp. 11–20. [Online]. Available: http://doi.acm.org/10.1145/1456455.1456458
10. G. Liang, X. DING, R. H. DENG, B. Xie, and H. Mei, "Remote attestation on function execution."
11. M. Nauman, T. Ali, and A. Rauf, "Using trusted computing for privacy preserving keystroke-based authentication in smartphones," *Telecommunication Systems*, vol. 52, no. 4, pp. 2149–2161, 2013.
12. A. P. Kosoresow and S. A. Hofmeyr, "Intrusion detection via system call traces," *IEEE software*, vol. 14, no. 5, pp. 35–42, 1997.
13. "Trusting Computing Group," http://www.trustedcomputinggroup.org/, 2014, [Online; accessed 19-July-2014].

14. P. A. Loscocco, P. W. Wilson, J. A. Pendergrass, and C. D. McDonell, “Linux kernel integrity measurement using contextual inspection,” in *Proceedings of the 2007 ACM workshop on Scalable trusted computing*, ser. STC ’07. New York, NY, USA: ACM, 2007, pp. 21–29. [Online]. Available: http://doi.acm.org/10.1145/1314354.1314362
15. C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang, “Remote attestation to dynamic system properties: Towards providing complete system integrity evidence,” in *Dependable Systems & Networks, 2009. DSN’09. IEEE/IFIP International Conference on.* IEEE, 2009, pp. 115–124.
16. S. Axelsson, “Intrusion detection systems: A survey and taxonomy,” Technical report, Tech. Rep., 2000.
17. H. Debar, M. Dacier, and A. Wespi, “Towards a taxonomy of intrusion-detection systems,” *Computer Networks*, vol. 31, no. 8, pp. 805–822, 1999.
18. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, “A sense of self for unix processes,” in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on.* IEEE, 1996, pp. 120–128.
19. M. Benantar, *Access control systems: security, identity management and trust models.* Springer Science & Business Media, 2006.