A Complete Behavioral Measurement and Reporting: Optimized for Mobile Devices

Toqeer Ali[1], Megat Farez Azril Zuhairi[1], Jawad Ali[2], Shahrulniza Musa[1], Mohammad Nauman[3]

[1] {toqeer@iu.edu.sa, megatfarez@unikl.edu.my, jawad2k3@gmail.com}
Malaysian Institute of Information Technology,
Universiti Kuala Lumpur
[2] Islamic University of Madinah,
Madinah, Saudi Arabia
[3] Max Planck Institute for Software Systems,
Germany,

Abstract. Security is an important factor in today's IT infrastructure due to complex and vast variety of malware threats. One way to tackle these malware is via signature-based techniques. However, this requires human effort in identification of threats and is not scalable. The second way is to detect malware via behavior-based reference monitor so called 'O-Day' malware. In this paper, we have optimized behavior-based tech-nique for a specific use-case, based on today's enterprise requirement. We have built behavior-based light-weight reference monitor to measure and report a complete system call sequences as well as its arguments. The measurements are stored into Trusted Platform Module (TPM) pro-tected location. The reference monitor splits the sequences of system calls and its arguments. Arguments and their verification is performed inde-pendent of each other via machine learning techniques. The behavior monitor is designed and developed on the core Linux Security Module (LSM). The same monitor is also designed and developed for Android-based platform via a newly built architecture called Android Security Module (ASM).

1 Introduction

The growing rate of attacks on software applications makes it infeasible to rely only on hash based detection techniques [6, 8, 31]. Similar is the case for large sets of applications running on remote devices which are even not in control of the corporate. For monitoring these remote applications, plenty of work has been carried out [10, 15, 17, 28, 30]. Software can also monitor and report behavior of the application to remote devices. However, on a compromised system, behavior monitor itself can be infected.

In this regard, Trusted Computing Group (TCG) has provided a specifi-cation, that provides a way to monitor remote applications known as *remote attestation.* Manufacturers have made co-processor on these specifications called TPM [1]. Based on this core, apart from hash based attestation [24] a number of dynamic behavior attestation techniques have also been proposed. The prob-lem with existing dynamic behavior attestation [2,13,26] based on TPM is that, some solutions suffer from performance of behavior locking in the hardware while others lack the ability to cover the broad range of malicious behavior [4, 27]. In this paper, we propose a solution to balance the performance problem as well as to report every aspect of applications behavior.

Additionally, existing solutions have been proposed for commodity hardware. However, there is a need for this solution to be shifted on mobile devices. Specifi-cally, Android has received immense popularity due to its open source nature. A large number of users shifted to use Android and it is worth mentioning that the numbers of malware writers are also shifted towards Android to get the secret information from mobile users, such as, credit card numbers or password etc. In this paper, we focused on Android OS to provide dynamic measurement and reporting of an enterprise application.

2 Use Case

Large number of enterprise applications are running on mobile devices, e.g,(CIMB Clicks). They are solely depending on the operating system's own secu-rity mechanism. Enterprises, don't have a mechanism in place to monitor their own application's behavior running on smartphones. The existing solutions, to report the behavior remotely, are either software-backed or even if they are hard-ware supported then they can report static behavior of the applications [21] only. Software-backed solutions are it-self vulnerable to attack in an effected operating system environment. The static techniques that can report static hashes are not even able to detect attacks that do not change code of the applications, such as, the infamous recent 'glibc' and 'Heartbleed' attack. The solution provided in this paper is for those enterprises that would like to monitor their own application's health on the remote smartphone devices. The solution measures and reports minimal though complete behavior of application running on IOT devices as well as smartphones.

The contribution of this paper is as follows:

- Monitored all the aspects of an applications behavior and measured according to the TCG specification.
- Optimized the state-of-the-art detection mechanisms that is suitable for software-based behavior monitor, however, not applicable directly in a *re-mote attestation* scenario.
- Modified attestation protocol for the behavior reporting and verification of a new technique.
- Designed architecture of an existing TCG-based reference monitor for *Mobile* platforms

Outline The rest of the paper is organized as follows: Section 3 we discuss both the state of the art techniques for enterprises as well as for android platform. In section 4 we provide our problem statement for the proposed research. Section 4 describes the design goals and Section 5 discuss proposed architecture in terms of Linux based behavior monitor in detail. Finally, in Section 6 an attestation architecture is designed for Android platform that can capture and report the behavior of an applications in a trusted way.

3 Remote Attestation: State of the Art

To verify either the complete operating system or target application, two ma-jor steps have to be performed in every remote attestation scenario. First, the static hashes or dynamic behavior should be measured and stored in the TPM. Secondly, on request, this behavior should be reported in a secured way so that it can be verified. For understanding of problem statement and our contribution we have shortly described the past work.

Initially, TCG has given solution to measure the boot process from BIOS till the boot-loader. The work being done in [24] has built the chain-of-trust from BIOS to kernel and kernel is able to measure the complete operating system including all the applications and configuration files. The problem with this work is that it is similar to hash-based intrusion detection, that is, it only takes hashes of every executable and stores it into the TPM's *Platform Configuration Register*(PCR). Although, a hash-based IDS does not have the capability to securely store hashes and verify at the remote system.

It has been realized that a corporate can be interested in a single application verification instead of complete operating system measurement and reporting. The work done in [16] has come with the solution to measure and report a single targeted application and all the related application that communicates with it. This work reduces the overhead, however, it only report the static hashes of the application and its information flow.

Apart from hash-based techniques, researchers have moved towards dynamic behavior attestation techniques. In the traditional OS environment, either on hand-held devices or PCs, system call is generated to request a resource or ser-vice. The work being done in [13] has measured every system call and reported to the remote party for behavior verification. However, according to our experi-ments it was not a feasible solution to implement. Because it creates bottleneck on the TPM while performing `SHA-1` hash and `PCR Extend` operations on every system call. Also, the solutions are not able to detect malicious behavior with system calls only.

Another approach towards DBA is LKIM [18] which tends to measure kernel data structures at runtime and represent graph to detect anomaly. The problem with this technique is measuring data structures at runtime creates a bottleneck on the client platform and sending this data to the remote platform is also an overhead on the network.

Some of the solutions directly took work being done as an IDS technique and implemented in remote attestation scenario [3], [19]. The problem with these techniques are that the solutions are implemented as it is in Linux kernel that were developed for Host-based IDS. Remote attestation differs in a way that it needs to securely measure behavior of the operating system as well behavior should be as minimal as a malicious behavior can be detected. However, these techniques only considered detection mechanism instead of performance issue related to remote attestation.

Remote attestation on smartphones: Nauman et al. [21] proposed and implemented an attestation mechanisms in Android security framework on both the operating system level and on top of Virtual Machine. The mechanism is based on two level of granularity i.e. Either Application level attestation or Class-level attestation. Both of these attestation required root-of-trust by im-plementing TPM hardware or emulator (software) on a device. As smartphones have lack of TPM hardware so the chain-of-trust is established by Mobile Trusted Module (MTM) according to TCG specification. To help-out the architecture for presence of a root-of-trust they have created a simplified lightweight TPM em-ulator that can only provide the functionalities of proposed solution. This will lead to low computation charges and battery consumption as much as possible.

After measurement by either application level or class-level the attestation token that contains *PCR Quote* and measurement logs are sent to challenger to check the state of remote smartphone. The challenger first validates TPM authenticity by verifying digital signatures included in the Quote. Afterwards, the measurement (hashes) of each loaded executable reported in the log is verified in database of known-good and known-bad hashes. This is the first and foremost adopted solution on the smartphones based on TCG specification, however, it takes static hashes of applications and classes running on Android OS. Thus, in this solution the mentioned motivating use case cannot be covered.

3.1 Android: State of the Art Software-based Behavior Monitors

CopperDroid: is a framework [23] that introduces an idea to analyze and classify low-level as well as high-level Android-specific behaviors. The technique retrieved behavior of an applications through system calls interception by IPC and RPC communication. CopperDroid is built over the famous emulator called QEMU that can automatically carry out analysis of dynamic behavior of malware run-ning on Android platform. The author enhances Android emulator that is able to track system calls. They have developed a behavior analyzer to verify these system calls. The reason to mention this work is that, it intercepts system calls at the kernel level and performs analysis. There should be a generic solution that can capture system calls at the kernel level so any technique can build their own reference monitor on top of that. Aurasium [29] is a novel and deployable technique as well as tool that enables dynamic and fine-grained policy enforce-ment of Android applications. To intercept relevant events, Aurasium operates only on a application level rather than interacting with system-level hooks or acquiring access to root and re-flashing device.

Similarly the *CrowDroid* is a framework that relies on malware detection systems in mobile devices [7]. Main idea in this framework is to dedicate a central remote server for analysis of applications running on mobile devices. This server is specifically responsible to characterize suspicious or normal activities of the client device. For this, the author developed Crowdroid which is a lightweight client which can easily be made available on Google Playstore. This application when installed on any mobile device is capable to monitor the behavior of ap-plication by collecting system calls using *strace* utility at user-space and send it to the central remote server. The remote server will then parse data and cre-ate a behavior profile on the basis of these system calls for further verification. More generally, the framework makes a remote verification mechanism between client and central server. However, the main problem with this solution is that, and which is of our interest, the behavior monitor is running in user-space and intercepting system calls vis *strace* utility. The problem is that, Crowdroid ap-plication itself is vulnerable to attack and malware may harm its integrity, thus we cannot rely on its behavior capturing and reporting. And software based solu-tions are more prone and vulnerable to several kinds of attack. More importantly their solution is a remote verification scenario, via which they are evaluating ap-plication. The standard way provided to verify remote application is via TCG's provided remote attestation solution. In short, a solution should be provided that instead of intercepting system calls in user-space it should be captured in the kernel level. Most importantly the behavior monitor itself should be trusted.

4 Problem Statement

The work being done in [15] has considered the performance issues related to remote attestation. According to them, they have measured *unique* sequence of system calls instead of every system call produced by an application. They have reduced the measurement and reporting log and performed verification-based automated machine learning modules. For further details reader can refer to [15]. Currently, this technique considers only system call to monitor and re-port, however, there are recent attacks that are based on arguments of system call instead of system call alone [22]. These attacks can go un-noticed by the ex-isting techniques. Now a solution is required that can measure unique sequence of system calls as well as unique *arguments* of system calls. The same problem can occur again if we measure and report every system call arguments, in result, it will be a bottleneck on the TPM (cf. Figure 1). In this paper, the author considered this issue to monitor and report sequence of system call as well as the arguments related to it. The existing behavior monitors are updated to split unique sequences of system calls and its related arguments. On the challenger side verification is performed on machine learning algorithms and the arguments are verified with standard *paths* of the target application.

The above mentioned solution is designed and developed on Linux kernel based on *Linux Security Module*(LSM). The reference monitor is developed to split sequence of system calls and its arguments and measured in the respected PCR. However, this solution is directly ported to the smarphones because of its architecture differences. Although, Android is based on the Linux kernel. One of the contributions of this paper is to embed existing PC-based behavior monitor into the Android Platform.

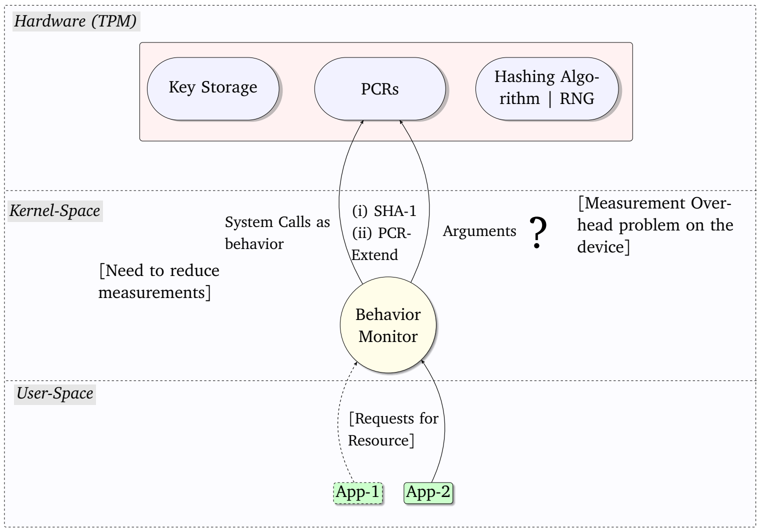


Fig. 1: Performance Problem While Measuring Every System Call and Its arguments

5 Design Goals

Following are the design goals which are essential to be fulfilled while designing the architecture.

1. *Efficiency:* The first goal of the proposed architecture is that it should be efficient. Main focus of the target architecture is to collect the parameters (also called as measurement of the parameters). The architecture should have the ability to measure these parameters in an *optimized* way.
2. *Dynamism:* As discussed earlier, the current techniques are efficient but the problem with them is that they cannot measure run time behavior of an application. Along with efficiency the other most important goal of the ar-chitecture is to measure dynamic behavior of application [25].
3. *Plugability:* The design should be pluggable at both the client and challenger ends.
4. *Testability and implementability :* The design of the attestation mechanism should be implementable and testable in the real environment.

5. *Reporting verses protection:* Generally, remote attestation in Trusted Com-puting provides reporting only as apposed to protection of target plat-form [24]. The design should be considered to report credentials securely to the challenger, so that if there is an increased rate of unknown behavior, the challenger should not provide any *secrete* information to this platform anymore.
Apart from the above, design goals are adopted, specific to attestation, from the guide provided by Justin et al. of the MITRE corporation in [9]. Justin et al. pointed out few general principles for attestation that ought to be satisfied while designing the architecture.

6. *Fresh Information:* The information about the target should be fresh and timely, so that it can well represent the running system.
7. *Comprehensive Information:* The attestation mechanism should be de-signed in such a way that it should collect detailed verifiable information about the target. Besides this, the measurement tool has to have access to internal state of target system. With this comprehensive information comes the issue of privacy as well as another issue that can be confronted i.e if an attacker gets the information that opens vulnerability for adversary. To cover this, the following objective is determined.
8. *Constrained Disclosure:* The targets platforms should enforce policies that can monitor and govern information/measurements sent to each challenger. The attestation mechanism should be able to identify the challenger. It can do so by many ways, for example, either it can ask the challenger to send some information strictly relating to it, so that the target can identify or it should send some secret code that would distinguish the challenger.
9. *Semantic explicitness:* The semantic content of attestation should be clearly defined. For example, how many measurements of semantic data are required for the challenger to identify behavior of target. Further, the content should be uniformed, so that it can determine in the form of logical functions.
10. *Trustworthy mechanism:* Finally, the attestation architecture should be de-signed in such a way that it assures trustworthiness to both the target and challenger.

6 Proposed Solution

As the future is going to be about the *Internet of Things* (IOT) and industry is rapidly moving towards it. As per Gartner research the very next year is going to see 5.5 million new devices going to be the IOT. The core operating system of these devices would be Linux. In short, IOT devices uses Linux as an operating system while Android also uses the same. And the market share of Android is the highest of all in smartphones. Keeping this thing in mind we designed and implemented the reference monitor for both, that is, for core Linux operating system as well as for Android. In the following sections we are describing the proposed solution.

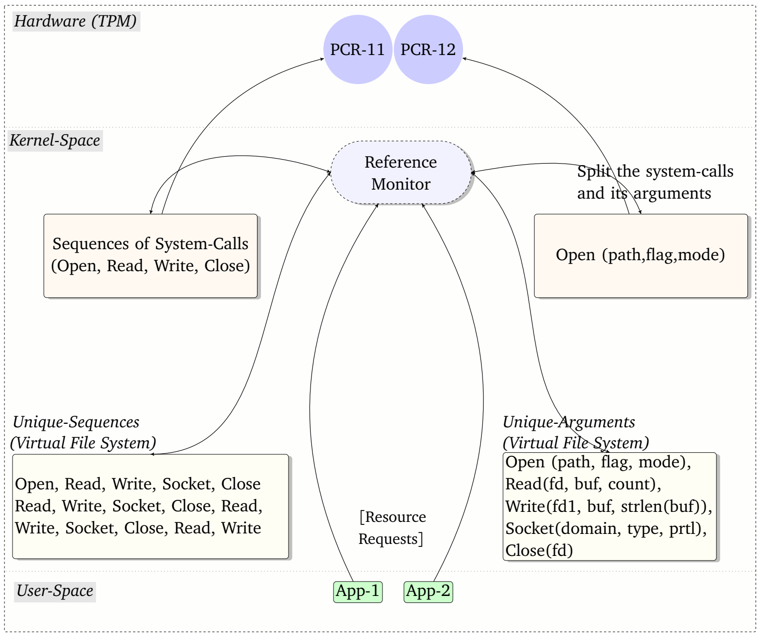

Hardware (TPM)
PCR-11
PCR-12
Kernel-Space
Reference Monitor
Split the system-calls and its arguments
Sequences of System-Calls (Open, Read, Write, Close)
Open (path,flag,mode)
Unique-Sequences (Virtual File System)
Open, Read, Write, Socket, Close Read, Write, Socket, Close, Read, Write, Socket, Close, Read, Write
Unique-Arguments (Virtual File System)
Open (path, flag, mode), Read(fd, buf, count), Write(fd1, buf, strlen(buf)), Socket(domain, type, prtl), Close(fd)
[Resource Requests]
User-Space
App-1
App-2

Fig. 2: Proposed Reference Monitor Designed For Efficient TPM Measurement

6.1 Linux-Based Behavior Monitor

In this paper we proposed a generic reference monitor for intercepting system-calls with arguments at kernel level. As shown in Figure 2, our reference monitor records behavior of applications. The operation of reference monitor is: When an application wants to access any resource, the *reference monitor* monitors ap-plications behavior and stores it. The behavior profile of an application is stored in two forms i.e. System-calls and its arguments. The system calls will be stored in form of fixed unique size windows in a virtual file system and their arguments will be stored separately in another virtual file system. Reference monitor will also be able to neglect or ignore repeated system with same arguments that was called previously. This way our framework will not be overburdened by the storage of same behavior again and again e.g. *open* called 5 times with same arguments, so it will be stored once in a log file. Furthermore, it is necessary to store the same system-call with different arguments in order to populate a strengthen profile for behavior of an application. Finally the reference monitor will also own an ability to store aggregated measurement of these system calls windows and their arguments in PCR. The measurement of system-calls win-dows will be stored in PCR-11 and their arguments will be stored in PCR-12. These PCR measurements along with logs (system calls log & arguments log) will further be used in verification at remote end.

Now, when a challenger would like to verify target application, it should send their credentials i.e. stored measurements in PCRs and measurement log. As we discussed earlier that system calls and its arguments are measured separately that can help to store complete behavior in the form of unique sequences as well as unique arguments. Furthermore the PCR-measurements must be truly signed by TPM key that can show its source authenticity. An attestation protocol is designed according to TCG specification that can ensure trusted communication between the end-nodes and verifier-nodes.

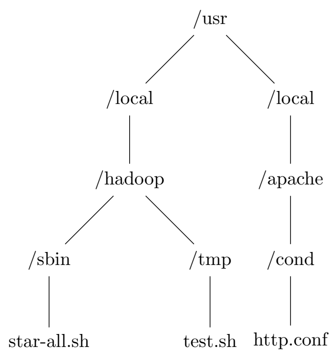
When challenger starts verification process, it will first verify TPM-signatures of the source. Afterwards, the PCR verification will be started that can prove integrity of data for any modification while transmission. After successful verifi-cation of PCR, the actual behavior of the application, that is, unique windows and its arguments are verified. The behavior dataset is fed into the trained classi-fier for normal and abnormal behavior. For more reading about machine learning that how it can identify normal or malicious behavior, we refer the reader to [15].

As stated earlier, the arguments of each system call is also stored in form of SML in a separate log file and its measurements are also stored in in TPM's PCR. The benefit of storing these arguments for behavior profile will prevent attacks which do not modify or alter the contents of system calls, however, just affects and causes anomaly in their arguments. Normally, every system call has four different types of arguments: pathnames, filenames, discrete numeric values and arguments used in execution of program. We investigate the arguments that are used more frequently in system calls i.e. pathnames and filenames. We build a separate verification model for each argument used by system call in the training phase. For this, we populate a normal behavior profile by a model *SyscallAnomaly* in an isolated environment on a challenger platform, known as *pristine profiling environment.* Afterwards, we denote the pathname of the files in cluster with a probabilistic tree structure, that contains the likelihood of all the involved directories along with their probability weights. For instance, in Figure 3 (a) the directory depth for *test.sh* is three, we note this tree as a normal activity for the particular case. While in Figure 3 (b) the directory depth for the same location is 4, we identify and store it into malicious class.

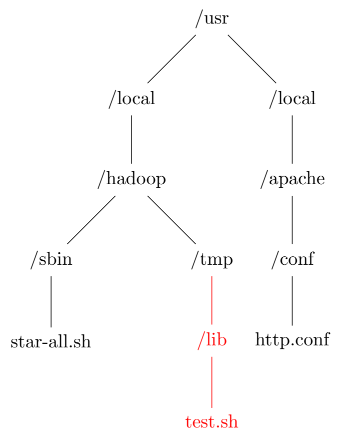
In order to identify arguments, the proposed architecture is able to verify applications profile of arguments along with their measurement sent by client end to the challenger end. In the first step PCR-measurements that contains the arguments information will be verified. After successful verification of PCR, the arguments log are matched in probabilistic models that were made during train-ing phase of behavior profiling. We specify a threshold value for each arguments in according to their directory depth for normal or anomalous.

6.2 Android-Based Reference Monitor for Remote Attestation

In the above section, we have introduced an architecture to measure complete and optimized dynamic behavior of application in terms of system calls and its arguments. The solution given is designed and built on top of LSM. The same solution is highly required for smart-phones as according to our literature, we



(a) Normal Tree



(b) Malicious Tree

Fig. 3: Normal and Malicious System Call Arguments

were unable to find such an architecture. Below we have discussed few solutions that intercepted system calls for behavior monitoring. Although, the solutions are not considering hardware-backed security. That motivated us to design and implement an architecture for Android platform.

As Android is one of the most popular among the mobile operating systems and because of its open-source nature we are enticed to design behavior monitor to verify corporate's applications remotely. The need of security in terms of malware detection is also of utmost importance. Malware detection tools and software programs are used for security in computer systems but they are not practicable on Android OS.

Different implementations in regard of Android security frameworks have been proposed [5, 11, 12, 20]. Each of them developed a security framework in different aspects according to their objectives. In particular, each framework inserts and retrieves same kind of hooks in Android. Furthermore, it is considered to be difficult for anyone to do changes inside Android security framework.

To resolve this issue, Android Security Modules (ASM) [14] framework pro-vides a standard programmable interface that allows users to define their own *behavior monitor*. The prime feature included in this framework is that: It pro-vides generic hooks that are pluggable to all the existing frameworks. If a user wants to specify reference monitor, so ASM-Bridge provides related hooks ac-cording to need. It is not required for user to change the internal framework while declaring reference monitor.

Based on ASM, we have built our behavior monitor that can capture the system call hooks in kernel and store into the related PCR. Also the same be-havior is stored in application level as SML (cf. Figure 4) measurement of the application.

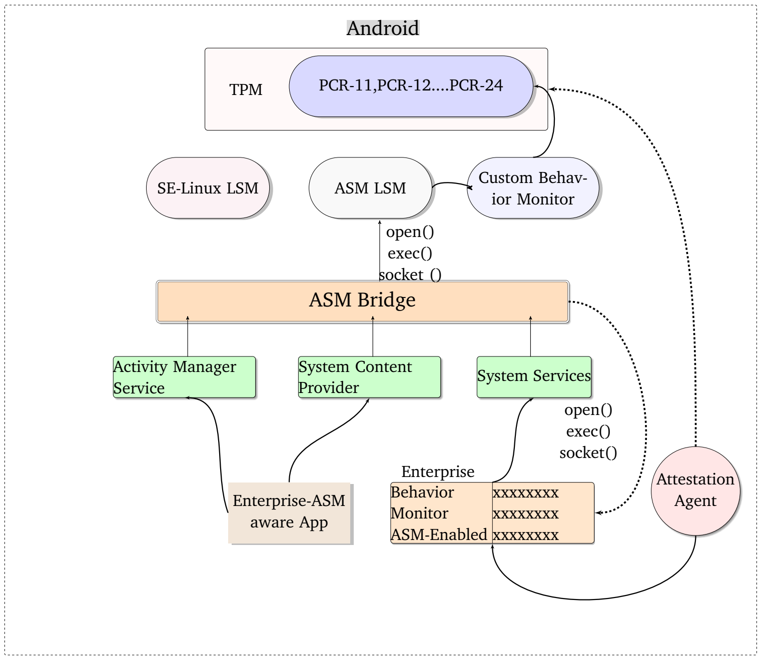


Fig. 4: Proposed Android Framework

Like mainstream Linux kernel where the system call behavior of an applica-tion is captured directly through LSM hooks and stored measurement of behavior in TPM. In our proposed Android architecture ASM-LSM provides a superset of these hooks that contain LSM hooks as well as Android specific hooks. On Android OS when application runs, the proposed framework captures behavior at two levels: *User-space* and *kernel-space*. ASM does not allow user to directly interact with ASM-LSM, for this ASM-Bridge is used as a mediator between kernel-space and user-space. The measurements of behavior will be stored in form of unique windows of specified size. The main purpose of using ASM-LSM is to communicate with TPM and ensure the chain-of-trust by enabling IMA [24]. As shown in Figure 4 the ASM-Bridge stores all log entries of each application behavior while their corresponding aggregated hashes are stored in TPM via our custom behavior monitor.

In order to perform attestation process, the attestation agent collects creden-tials i.e. PCR-Quote and log of system calls and its arguments. As we provide detail of attestation mechanisms regarding system calls and arguments in Section 6.1, so we implement similar mechanism to the Android architecture. An attes-tation protocol is developed that can ensure trusted state of the device. Upon a challenger's request for verifications, the framework starts preparing a response for verification. The response contains final measurement from PCR along with their log. The challenger verifies credentials and knows state of platform whether it behaves in trusted manner or malicious.

7 Conclusion

Mobile and IOT devices are emerging and the applications running on these devices are vulnerable to various attacks. A plenty of buffer-overflow attacks are there, such as, 'glibc' and 'Hearbleed' that actually do not change code rather the behavior is changed. Many solutions have been presented to monitor system calls as behavior of application. The solution of this paper is to efficiently and securely report behavior to remote platform. Moreover, major contribution of this paper is two fold. Firstly, a Linux-based behavior monitor is enhanced to measure system-call arguments along with the system calls. Secondly, reference monitor has been modified for Android platform. An ASM-LSM aware application has been developed and tested in the Android emulator while verification of behavior is out of scope of this paper.

References

1. "Tcg. trusted computing group, http://www.trustedcomputinggroup.org/."
2. T. Ali, M. Alam, M. Nauman, T. Ali, M. Ali, and S. Anwar, "A scalable and privacy preserving remote attestation mechanism," *Information-An International Interdisciplinary Journal*, vol. 14, no. 4, 1193–1203(2011).
3. T. Ali, M. Nauman, and X. Zhang, "On leveraging stochastic models for remote attestation," in *Trusted Systems.* Springer, 290–301(2011).
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, 407–421(2016).
5. A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "Mockdroid: trading privacy for application functionality on smartphones," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications.* ACM, 49–54(2011).
6. A. Bianchi, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Blacksheep: detecting compromised hosts in homogeneous crowds," in *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 341–352(2012).
7. I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices.* ACM, 15–26(2011).
8. D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "A quantitative study of accuracy in system call-based malware detection," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis.* ACM, 122–132(2012).
9. G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. OHanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, 63–81(2011).

10. Z. Dawei, H. Zhen, J. Yichen, D. Ye, and L. Meihong, "Protocol for trusted channel based on portable trusted module," *Communications, China*, vol. 10, no. 11, pp. 1–14, 1-14(2013).
11. M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D. S. Wallach, "Quire: Lightweight provenance for smart phone operating systems." in *USENIX Security Symposium*, vol. 31, (2011).
12. W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, p. 5, 5(2014), publisher=ACM.
13. 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, 11–20(2008).
14. S. Heuser, A. Nadkarni, W. Enck, and A.-R. Sadeghi, "Asm: A programmable interface for extending android security," in *Proc. 23rd USENIX Security Symposium (SEC14)*, (2014).
15. R. Ismail, T. A. Syed, and S. Musa, "Design and implementation of an efficient framework for behaviour attestation using n-call slides," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication.* ACM, 36(2014).
16. 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, 19–28(2006).
17. M. LeMay and C. A. Gunter, "Cumulative attestation kernels for embedded systems," *Smart Grid, IEEE Transactions on*, vol. 3, no. 2, 744-760(2012).
18. 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, 21–29(2007).
19. F. Maggi, M. Matteucci, and S. Zanero, "Detecting intrusions through system call sequence and argument analysis," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 4, 381–395(2010).
20. M. Nauman, S. Khan, and X. Zhang, "Apex: extending android permission model and enforcement with user-defined runtime constraints," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security.* ACM, 328–332(2010).
21. M. Nauman, S. Khan, X. Zhang, and J.-P. Seifert, "Beyond kernel-level integrity measurement: enabling remote attestation for the android platform," in *Trust and Trustworthy Computing.* Springer, 1–15(2010).
22. C. Parampalli, R. Sekar, and R. Johnson, "A practical mimicry attack against powerful system-call monitors," in *Proceedings of the 2008 ACM symposium on Information, computer and communications security.* ACM, 156–167(2008).
23. A. Reina, A. Fattori, and L. Cavallaro, "A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors," *EuroSec, April*, (2013).
24. R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a tcg-based integrity measurement architecture."
25. H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," in *Proceedings of the 14th ACM conference on Computer and communications security.* ACM, 552–561(2007).

26. 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, 51:1–51:7(2012).
27. T. A. Syed, S. Jan, S. Musa, and J. Ali, “Providing efficient, scalable and privacy preserved verification mechanism in remote attestation.”
28. C. Wang, C. Liu, B. Liu, and Y. Dong, “Div: Dynamic integrity validation framework for detecting compromises on virtual machine based cloud services in real time,” *Communications, China*, vol. 11, no. 8, pp. 15–27, 15-27(2014).
29. R. Xu, H. Saïdi, and R. Anderson, “Aurasium: Practical policy enforcement for android applications.” in *USENIX Security Symposium*, 539–552(2012).
30. W. Xu, X. Zhang, H. Hu, G.-J. Ahn, and J.-P. Seifert, “Remote attestation with domain-based integrity model and policy analysis,” *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 3, 429–442(2012).
31. H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, “Panorama: capturing system-wide information flow for malware detection and analysis,” in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 116–127(2007).