POSTER: DroidShield: Protecting User Applications from Normal World Access

Darius Suciu
Stony Brook University
New York, USA
dsuciu@cs.stonybrook.edu

Radu Sion
Stony Brook University
New York, USA
sion@cs.stonybrook.edu

ABSTRACT

Smartphones are becoming the main data sharing and storage devices in both our personal and professional lives, as companies now allow employees to share the same device for both purposes, provided the company's confidential information can be protected. However, as history has shown, systems relying on security policies or rules to protect user data are not airtight. Any flaw in the constructed rules or in the code of privileged applications can lead to complete compromise. In addition, we can not rely only on TrustZone[1] world separation to isolate confidential data from unauthorized access, because in addition to severe limitations in terms of both communication and memory space, there is a very low limit on the number of applications that can be installed in the secure world before we can start questioning its security, especially when considering code originating from multiple sources. Thus, the solutions currently available for TrustZone devices are not perfect and the data confidentiality can not be guaranteed. We propose an alternative approach, which involves providing the majority of secure world application advantages to a set of normal world applications, with almost none of the drawbacks by relying only on the TrustZone world separation and the TZ-RKP[2] kernel protection scheme.

Keywords

ARM TrustZone; data protection; mobile device security; secure execution

1. INTRODUCTION

The fast evolution of multi-functional mobile devices has made them a priority target for attackers. Smartphones are used not only for communication and entertainment, but now companies provide work phones for their employees to store passwords, emails and other confidential information. This trend motivates attackers to target these devices. The attacks range from simple attempts to install exploits designed to get unauthorized access by taking advantage of kernel vulnerabilities, up to attempts by specialized agencies to snoop private data by employing costly brute force decryption processes. In consequence, a set of defensive measures have been deployed to help protect the data/applications deemed sensitive and worth protecting. Depending on the technology deployed on a specific phone, users can use the TrustZone techonology to build a secure environment with its own operating system and applications, or to a component like Apple's secure enclave[4], that can safely encrypt their data. On TrustZone enabled devices, users can isolate sensitive applications in a secure environment, reachable only by a limited set of predefined commands/messages. This technology allows users to separate the everyday activities, like browsing the web and downloading entertainment applications from the Play/Apple Store, from critical operations such as entering a password, authorizing a payment on the credit card or sending confidential emails. By using the scheme proposed in SeCReT[8], we can verify the identity of parties involved in the communication between environments and encrypt their messages. However, even though TrustZone offers us an environment that cam be used to protect confidential information, we have no guarantees this information will not be eventually leaked to attackers. There exist several confirmed attacks that allowed unauthorized normal world users to obtain not only access, but even control over the secure world operating system (OS).

In this paper we propose a new application protection scheme, which can leverage the secure world to shield normal world confidential data from unauthorized access that might originate not only from user applications, but the normal world kernel itself.

CCS'16 October 24-28, 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4139-4/16/10.
DOI: http://dx.doi.org/10.1145/2976749.2989052

2. PROBLEM DESCRIPTION

Lets consider a TrustZone enabled Android phone, used for both work and recreational activities. Usually it has a plethora of normal world applications installed, and some malicious code could infiltrate the system, as not all applications are checked for vulnerabilities or suspicious behavior before installing. Since the normal world kernel contains thousands lines of code, which change after each update, we cannot guarantee it does not contain any exploitable vulnerabilities. An attacker could compromise a application and use it to gain root access to the normal world kernel. This would allow him to issue commands to secure world applications, and since all communication between environments is done though commands issued by the normal world kernel, he can intercept and control the communication between secure world applications and the external world. Moreover, experience has shown that not all code checked by Original Equipment Manufacturers (OEMs) is free of vulnerabilities, allowing attacks like [3] to use small exploits to get control of the secure world kernel. Once an attacker has control over the secure world, no information on the device can be protected. In other worlds, the introduction of third party code pollutes the secure world and renders all the measures taken to ensure data confidentially useless.

There is an intuitive solution to this problem. Keep only a tiny OS in the secure world, whose size would allow a complete check for vulnerabilities, and create two categories of applications in the normal world isolated from each other. An existing approach, Samsung Knox[6], aims to create an isolated environment for work related applications and seems to solve our issue. However, due to the relaxed separation model relying on SEAndroid[7] security policies to ensure the separation of Knox and user applications, a set of vulnerabilities have been discovered and presented in [5], indicating that additional work is required for a complete solution. Moreover, some applications need to be protected from unauthorized kernel access too. For example a Digital Rights Management(DRM) and other entertainment applications need to be protected against users with root access to the normal world kernel, has they might try to bypass the limits/rules imposed by the content provider. Thus, we propose a new design for TrustZone enabled devices, in which the secure world is only used to ensure the integrity of the normal world and that applications can run in an environment protected from all unauthorized normal world access.

We present the decision process that should be taken every time we install a trustlet (application containing sensitive data) in Figure 2. As can be seen, the currently available technologies only allow users to protect a trustlet by installing it in the secure world, but I/O drivers might not be available and need to be constructed. Additionally, the trusted computing base would automatically be extended, as now the trustlet and driver's code, verified by only OEMs would be part of the privileged secure world environment. However, by integrating our solution in the TrustZone environment, trustlets can be installed in the normal world instead, avoiding all these problems.

3. SYSTEM OVERVIEW

In order to protect application sensitive data from unauthorized entities, we have to consider all the application states (running, stopped, descheduled), in each state data being stored in memory, on disk or in CPU caches. By giving each application a unique key and relying on an powerful encryption service, we can ensure that data stored on disk is only accessible to applications that have a corresponding key associated in the secure world. To ensure data cannot be leaked or altered when cached, we just have to flush all the registers and caches on context switch events that involve our target applications, similar to switching between secure and normal world contexts. The most difficult data location to protect is the memory, as flushing or encrypting all the data stored there is not an option.

Unfortunately TrustZone does not offer hardware capabilities to create multiple containers for user application, but we can leverage it to create our own protected space. Consider a TrustZone enabled device, with Secure boot enabled to verify both the normal world and secure world kernel code integrity during boot and the TZ-RKP solution deployed,

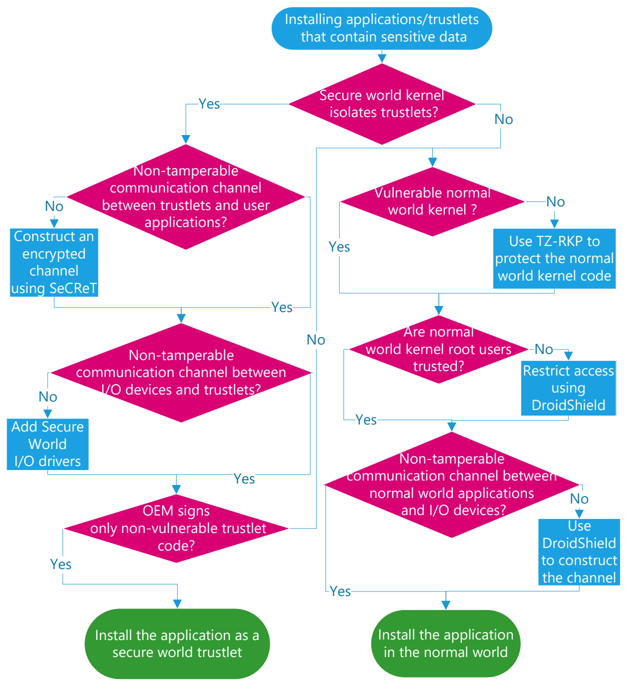
Installing applications/trustlets that contain sensitive data
Secure world kernel isolates trustlets?
Yes
No
Non-tamperable communication channel between trustlets and user applications?
No
Construct an encrypted channel using SeCReT
Yes
Vulnerable normal world kernel ?
No
Yes
Use TZ-RKP to protect the normal world kernel code
Non-tamperable communication channel between I/O devices and trustlets?
No
Add Secure World I/O drivers
Yes
No
Are normal world kernel root users trusted?
No
Restrict access using DroidShield
Yes
OEM signs only non-vulnerable trustlet code?
Yes
Non-tamperable communication channel between normal world applications and I/O devices?
No
Use DroidShield to construct the channel
Yes
Install the application as a secure world trustlet
Install the application in the normal world

Figure 1: Decision process used when installing an application that contains confidential data

which guarantees the integrity of normal world kernel code during runtime. Since there are no user applications in the secure world in this design, we only need a very limited set of secure monitor commands (SMC) to communicate with the secure world, and none should originate from user applications. This means only approved kernel functions have access to the secure world, moving it out of attacker's reach. Moreover, we can leverage TZ-RKP to protect the integrity of all installed trustlets's code.

Since we can trust the secure world, we can use it to build the containers that protect our confidential information from normal world access. Lets consider a small set of special user applications (trustlets), consisting of a TIMA Keystore and a generic DRM application, that need to be isolated from unauthorized normal world access. Keystore is a trustlet used to install and provide encryption keys, at the request of treads running within the *system_server* or process with a system unique identifier (UID), while DRM needs to decrypt in real-time data belonging to an outside content provider.

In order to build a container for each trustlet, we maintain a list of all trustlets in the secure world, which allows us to detect when one of them is scheduled and perform the operations necessary to guarantee its security. Each entry in the list has an associated key stored alongside it and a pointer to a table containing memory page permissions, as depicted in Figure 3. Each entry in this table points to an entry in the normal world page table, whose permission bits it controls. After a trustlet reserves a memory page, both the write and read bits for the respective entry in the normal world page table are set by the secure world kernel to values contained in the permission's table entry. Because we use the TZ-RKP to protect memory pages, the normal world can't access pages denied by the secure world kernel.

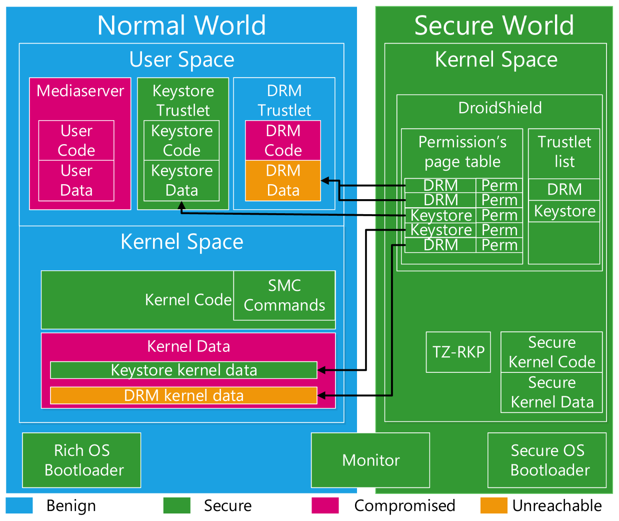

Figure 2: Framework of a system using DroidShield

Depending on the user preferences, the secure world kernel can either prevent the normal world kernel from swapping out reserved pages, showing memory portions to the normal world as temporarily unavailable, or it could save the page content in secure world memory, clearing the page to be used by normal world application, and restoring its state from the secure memory when the trustlet is rescheduled. The first option is more viable for user space data pages, as usually they do not need to be shared among applications. However, memory-mapped devices can allocate kernel data pages, which might need to be shared between multiple applications, a situation more suitable for swapping out memory pages in the secure world. Either way, using this design, the secure world can ensure that all user data can be protected also while being stored in memory.

By locking all data used by a trustlet, we can effectively isolate it from all normal world access. However, sometimes applications need to communicate with each other and share data. For example, Keystore can only provide the keys to other applications if we construct a communication channel between them. This is a problem easily solved in this design, as we only have to share memory pages between trustlets, a function easily accomplished by adding multiple entries in the Permissions Page Table pointing to the same normal world page table entry, one for each trustlet involved in the communication. Shared pages imply fast communication, as data is accessible to both applications without any overhead.

For example, lets consider scheduling a Keystore trustlet in the normal word kernel after a DRM application consumed its allocated time slice. In Figure 3 we can see the system's state while the Keystore is running. All DRM data is locked away and marked as unreachable, while the data specific to the Keystore itself is considered secure, as it is only accessible while the trustlet is running. Neither the MediaServer application or kernel threads can access the data of either trustlets. This model does not involve adding secure world application and no extra drivers need to be written, making it highly scalable, as the installation of a trustlet does not have an impact on the secure world integrity.

There are some compromises made to allow trustlets to live in the normal world, which include the addition of a vulnerability to DMA attacks, as the model uses page table permission to control all virtual memory accesses, and a penalty in context switch performance, as any memory page not marked as non-swappable and reserved by multiple containers has to be backed up in the secure world and wiped. In the future we plan to investigate the impact on system performance and user experience, analyze the viability of using this solution in real-world mobile systems.

4. CONCLUSIONS

The current state of the art solutions in Trustzone security either cannot ensure that unauthorized entities can't access application confidential data or are not scalable in real-world scenarios. Thus, if we want to isolate the application sensitive data from all normal world access, we need to design a new protection scheme. In this poster we present a new TrustZone based solution that can protect all sensitive data contained in memory pages, allowing us to create individual application containers, which are isolated from all normal world access. This allows us to grant these applications the advantages previously available only to secure world trustlets. In contrast to existing approaches, we only rely on verifiable code and only the secure world is used to create and maintain the containers, protecting the application from even users that have normal world root access. Additionally, a page sharing mechanism allows us to create a fast communication channel between any user applications.

Acknowledgments

This work was supported by the National Science Foundation through awards 1161541, 1318572, 1526102, and 1526707.

5. REFERENCES

[1] ARM. Bulding a secure system using trustzone technology. *ARM Technical White Paper*, 2009.

[2] A. M. Azab, P. Ning, J. Shah, Q. Chen, R. Bhutkar, G. Ganesh, J. Ma, and W. Shen. Hypervision across worlds: Real-time kernel protection from the arm trustzone secure world. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 90–102, 2014.

[3] CVE-2016-2431. Online at https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-2431.

[4] M. Gulati, M. J. Smith, and S.-Y. Yu. Security enclave processor for a system on a chip. *U.S Patent No. 8,832,465*, September 2014.

[5] U. Kanonov and A. Wool. Secure containers in android: the samsung knox case study. *arXiv preprint arXiv:1605.08567*, 2016.

[6] SAMSUNG. Whitepaper: An overview of the samsung knox platform. November 2015.

[7] S. Smalley and R. Craig. Security enhanced (se) android: Bringing flexible mac to android. *NDSS*, 310:20–38, 2013.

[8] J. J. Soo, S. Kong, M. Kim, D. Kim, and B. B. Kang. Secret: Secure channel between rich execution environment and trusted execution environment. *NDSS*, February 2015.