Introducing Trust into the Digabi Platform

Samuel Laurén
Department of Information Technology
University of Turku, Finland
Email: samuel.lauren@utu.fi

***Abstract*—The use of students' personal devices makes properly securing electronic matriculation examination a particularly difficult challenge. How to make sure that the examinees do not have access to unauthorized materials, when they have unlimited physical access to the hardware?**

In this proposal, we provide an overview on how trusted computing based techniques could be applied to prevent attacks where the attacker has physical access to the hardware. We consider the use of Trusted Platform Modules (TPM) and the Integrity Measurement Architecture (IMA) for remote attestation in the context of DigabiOS.

***Index Terms*—trusted computing, remote attestation, trusted platform modules**

I. Introduction

Traditionally, computer security has focused on attack scenarios where the attacks are originating from a malicious external adversary. Many security techniques and defenses have been shaped by this assumption. Scenarios where the "adversary" is the user themselves, have received little attention in comparison.

One of the principal attacks affecting the Digabi platform has to do with the examinee getting access to additional resources not approved by the exam organizers. Currently, the DigabiOS uses various security measures trying to prevent the user from eg. installing additional software packages or mounting additional storage devices. While these security mechanisms are effective, they are still dependent on the integrity of the operating system and the underlying hardware platform.

One possible attack that has been proposed [1] is based on booting the DigabiOS in a virtual machine instead of real hardware. Since the presence of a properly functioning hypervisor is, by design, mostly hidden from the operating system that is running on top of it, the guest operating system has limited ability to notice that it does not have the full control over the hardware. In this attack, the student could simply boot the examination OS as a guest and simultaneously have the ability to access additional material from the host. What is more, the hypervisor could conceivably alter the operation of the guest OS since it has unrestricted access to the contents of its memory. This type of attack could undermine most other security techniques.

Trusted Computing techniques have emerged as a way to introduce trust into computing. In this context, trust can be understood as the ability to have systems' behave in an expected way. In a general sense, the problems of trust come down to figuring out where the trust should be anchored. In the previously presented hypervisor-based attack, the security of the system dependent on the security of the operating system kernel, by undermining the operating system security "from the outside" the attacker was able to undermine the security of the entire system.

Trusted Platform Modules (TPM) [2] provide a way to anchor trust in hardware. As a separate, dedicated cryptographic co-processors, TPM chips are automatically protected from whole classes of software-based attacks, effectively raising the bar for adversaries.

In this proposal, we describe how Linux's Integrity Measurement Architecture [3] could be used to provide remote attestation for DigabiOS. Remote attestation could provide the system with the ability to collect cryptographic proofs that the connected clients are, in fact, executing the unmodified DigabiOS and that the system is not affected by any other piece of software.

In section II, we provide a brief overview of TPM chips and the functionality they provide. In section III, we introduce the concept of remote attestation and discuss how it can be used to verify the state of remote system. In section IV, we consider how remote attestation capability could be implemented in DigabiOS using existing technologies. In section V, we image how the attestation could work from examinee's point of view. In section VI, we discuss the challenges associated with out approach and how they could be solved. Finally, in section VII we consider how applying the proposed techniques would improve the security of DigabiOS.

II. Background

Trusted Platform Modules are dedicated chips with the ability to securely create and store cryptographic keys, perform operations using those keys, and keep a limited amount of information about system state. Due to the limited interface exposed by these chips, they are rendered immune to a number of software based attacks. The keys never leave the chips unencrypted to be exposed to a malicious adversary.

Each TPM chip has a number of different keys associated with it. Endorsement Key (EK), is an asymetric, unique and immutable key that is bound to the TPM at the time of manufacturing. Additionally, each TPM can have a number of Attestation Identity Key that are used for attestation.

One of the most interesting abilities of these chips is their set of Platform Configuration Registers (PCRs). What makes PCRs useful is the limited interface they offer. PCRs do not

support arbitrary assignments, but instead their value can be changed through hash extend operation as seen in Equation 1.

$$Extend(n, X) : PCR_n \leftarrow H(PCR_n||X) \quad (1)$$

Extending PCR_n with X takes the current value of PCR_n, concatenates it with X, and hashes the result using H before assigning it into PCR_n. Additionally, for some PCRs, their value can only be reset upon reboot.

More powerful features can be built on top of this foundation. TPM-based Measured Boot mechanism makes it possible for systems to prove that they are executing the piece of software they claim they are. Measured Boot depends on a secure Root of Trust for Measurements (RTM). Typically a TPM enabled system includes a component that acts as a Static Root of Trust for Measurements (SRTM). Before handing the control to the operating system loader, this Core Root of Trust Measurements (CRTM) hashes a number of critical system components, including itself, firmware, and the OS loader that is about to be executed, and saves this information into a set of PCRs using the hash extension mechanism.

After CRTM, the operating system loader can then, in turn, hash the kernel and the initial root file system and store the information in the PCRs. In the same way, the kernel can then hash the included applications. In the end, the hash chain will store traces of the entire software stack of the machine.

The CRTM does not prevent the machine from booting regardless of the hashes it collects, all it does is provide the evidence via TPM for later inspection.

III. Remote Attestation

Measured Boot mechanisms makes it possible for remote parties to attest that a the client truly is executing the software it claims it is. Quotes are the critical piece of TPM functionality making this feat possible.

Quotes are signed reports that can contain a summary of the state of a set of PCRs along with a nonce. Software can ask the TPM to generate a quote based on a set of registers and an arbitary value. The TPM combines the state of the PCRs and the nonce and signs it with a key that only it has. Knowing the expected value of the PCRs and the nonce, the receiver of the quote can be certain that the system is in the expected state.

This basic attestation scheme is only dependent on the security of the TPM and the RTM. The secret keys are only known to the TPM but anyone with access to the public part of the signing key can verify that the quote originated from the machine with that TPM.

IV. Implementation

The presented mechanism conveys the idea behind TPM-based remote attestation. It provides a glance into the world of trusted computing and how one can extend trust from a small trusted component to cover an entire system.

While it is possible to build a working prototype on top these primitive operations offered by TPM chips, it is not necessary since many of the components have already been developed and battle-hardened by real-world use.

In this section, we consider existing components that could be included into the DigabiOS in order to enable remote attestation.

A. Trusted GRUB

Trusted GRUB [4] is a fork of the ubiquitous GRUB [5] boot loader. It extends the original solution by implementing TPM-based measured boot functionality, allowing measuring the kernel.

B. Linux Integrity Subsystem

In the context of Linux, the principal facility for measuring and recording the state of the system is the aptly named Integrity Measurement Architecture [3] (IMA). This component has been included in mainline Linux kernel releases since version 2.6.30 in 2009.

Linux Integrity Subsystem enables the hashing of file system contents and safe recording of the collected hashes in extended file system attributes. Additionally, if a TPM chip is present, it can be used to store an aggregate value of the hashes using the hash extend method. This aggregate value can then be used for attestation.

C. Strongswan

Strongswan [6] is an open source IPSec-based VPN implementation. Aside from its VPN capability, Strongswan supports Trusted Network Connect (TNC) technology for TPM-based remote attestation [7]. Their solution has been used, for example, to provide mutual attestation in teleconferencing system [8].

While the primary purpose of their implementation is to control access to networks, Strongswan could be used just for its ability to do remote attestation. Trusted Computing Group has created a set of specifications for doing remote attestation.

For integrity measurements Strongswan can work with IMA to collect evidence of the systems state [9].

V. Example Flow of Events

In this section, we consider what a hypothetical deployment might look like from examinee's perspective.

1) Well in advance, the examinee tests that they can successfully boot the DigabiOS on their hardware. This process might include modifying the boot priority and secure boot settings. As a new addition, the examinee is encouraged to enable the TPM support in case it is disabled.
2) As a part of the compatibility test, the machine is also registered with the examination system. This does not alter the machine in any way but shares the public key portion of the machines Endorsement Key. This step is important for the integrity of attestation and should be done under supervision. Alternatively, the this step could create a new Attestation Identity Key and share its public portion with the system.

3) Additionally, as a part of the registration process the server records state of the Platform Configuration Registers associated with the firmware and the loaded DigabiOS kernel. Recording this state is important because it allows the server to later construct the correct values for the hash chains.
4) When the examination is about to start, the machine is booted into the DigabiOS. The CRTM records the state of firmware, configuration, and the boot loader. The boot loader measures the operating system and the operating system measures the applications.
5) The examination server asks the client machine to prove its state. The server sends a nonce and the client asks the TPM to produce a quote using the nonce and a given set of PCRs.
6) The server receives the quote and verifies that the signature was made with a known good Attestation Identity Key. The server also checks that the returned aggregate value of the state of the PCRs matches the expected one. This is somewhat machine specific since it is dependent on the firmware.
7) The machine is now attested by the server and the examination can begin.

VI. Challenges

While Trusted Platform Modules bring many benefits, they also introduce a set of challenges that need to be properly addressed. In this section, we consider some of the challenges associated with TPM applications.

A. Key Management

Key management is a critical part of the attestation process. For the server to be able to verify the quotes send by the client being attested, it has to have access to the corresponding public key.

For attestation to work, the machines used during the examination would have to be registered beforehand. This registration would include associating the public key part of a TPM-held AIK with the system.

It is already necessary for the examinees to test the DigabiOS before the actual examination date, to make sure that the machine can successfully boot the system and that the operating system is compatible with the hardware configuration. Considering this situation, adding another step of registering the TPM into the system does not seem unreasonable.

On a practical level, this could be done from within the DigabiOS using an automated tool. From the user's point of view, this would not require any extra work.

From a security perspective, this registration step would be the most important one, since it would establish a relationship between a given machine and the server. Care would have to be taken to avoid adversary from registering a malicious machine. For example, a virtual machine with a software-based TPM capable of disclosing its secrets would weaken the security of the system.

However, this registration step could be done well in advance under appropriate supervision to decrease the possibility of cheating.

B. Configuration Management

Traditionally, one of the main challenges of hash-based systems has been managing the set of valid hashes. With updates and installation of additional software, keeping track of the platform configuration can be difficult.

However, it can be argued that DigabiOS is in a good position to face these challenges. As a special purpose system, DigabiOS does not have to deal with updates and installations of arbitrary software. Once the final operating system image is produced, the hashes will remain unchanged. Additionally, the current deployment model already requires careful management of the client and server versions: the client has to be compatible with the server.

Because our ability to prevent virtualization based attacks is dependent on our ability to attest the integrity of the whole boot chain, we have to store measurements from the firmware. Since these are machine specific, they have to be recorded for all machines used for examination.

This undeniably complicates the deployment process. However, we believe that this step could be made mostly automatic. During the machine registration, the system would be booted with the same kernel and initial file system and the measurements stored on the server.

Applying BIOS and firmware updates after the registartion but before the final examination would pose a problem for our design. Once the platform has been registered with the system, any change in BIOS would be regarded as tampering and invalidate the hashes. However, such updates are often quite rare and the effects of this limitation could be, in practice, limited.

C. Machines without a TPM

The variety of examinees' machines poses a challenge for any kind of technical solution. Not all machine will have a TPM capabilities available. However, in order to decide what kind of policy would best fit the situation, one has to have an idea about the prevalence of TPM hardware.

Trusted Platform Modules are commonplace in newly sold laptops. In fact, Microsoft's minimum hardware requirements [10] for Windows 10 mandate that a TPM 2.0 compliant chip must be present and enabled by default on desktops. While these guidelines are recent, TPMs have long been prevalent in the PC market. In 2007, over 100 million TPM enabled computers were sold [11].

Taking these trends into account, one can expect that Trusted Platform Modules will become even more ubiquitous in the coming years.

However, at present there is likely to be a sizable portion of the users without a TPM or with a TPM disabled. For the later category, the TPM could be enabled through BIOS. While arguably arcane for the average users, one has to consider that

manual BIOS configuration is already necessary in many cases for the machine to be able to boot from the supplied USB drive.

Machines without a TPM could still be allowed to be used, but the fact that the machine has not been fully attested could be recorded on the server. In the case of potential problems, further analysis could be directed to the subset of machines where there is no strong cryptographic proof of authenticity. In addition, knowledge about the presence of TPM chips could be used to direct more supervision to the examinees whose machines have not been verified.

It is important to note, that whatever policy is enacted it has to be kept up to date with the current hardware trends. At present, it might be very well be reasonable to allow unattested clients to be used. However, if the present trends continue and the number of machine without a trusted platform module fades into a minority, this policy could be revised. At some point, it might be practical for students who do not have a complying machine to borrow a machine for the examination.

VII. Discussion

We began this proposal by considering what could be argued to be one of the main threats faced by the Digabi system. When the adversary is also the owner of the machine, the options available to defenders are scarce. How could one be certain that the machine is truly executing the unmodified DigabiOS?

Trusted Computing and the TPM chips in particular, offer a text book solution for this problem. Trusted Platform Modules offer a root of trust in an otherwise untrusted environment. This tiny kernel of trust can act as a foundation for verifying the state of an entire system. All in a way that can be automatically checked by an remote system.

Aside from the direct benefits of employing a measured boot type solution, it enables a whole set of orthogonal security mechanisms that could not have been implemented without it.

For example, once the integrity of the operating system has been verified, it can be trusted to provide correct monitoring data to remote parties. The DigabiOS could include a probe that continuously introspected the system to detect any runtime attacks and reported this data to the server. Without measured boot, any monitoring data provided by such probes would be suspect to malicious fabrication. With TPM-based measured boot in place, the attestation could be expanded to from file hashes to interactive properties of the system.

VIII. Conclusions

In this proposal, we have presented a plan for utilizing Trusted Platform Modules as a part of the Digabi system. By making use of the hardware already present in many laptops, the Digabi system would have a way to prove that the examinees are, in fact, running the unmodified DigabiOS while taking their matriculation exams.

In its current state, the trust placed on the Digabi system is somewhat misplaced. The system simply has no way of knowing that it has not been tampered with. All the other security measures built on top of the system are fundamentally limited by this lack of trust. This is acknowledged by the developers: an attacker could modify the USB drive containing the DigabiOS to include additional tools or relax its rules, or they could execute the whole operating system within a virtual machine.

Applying TPM technology would solve these problems and make all the other security features more effective. Where other security mechanisms are often dependent on the validity of the operating system kernel. Trusted Platform Modules anchor the trust to a hardware based solution.

While the technology is still relatively recent, there are well-established solutions within the Linux ecosystem for providing the kinds of attestations described in this paper. Not only is remote attestation possible, it can also be practical.

Hash-based remote attestation is not a silver bullet. Even with remote attestation in place, runtime attacks could still be possible against the system. Like almost everything in computer security, Trusted Platform Modules are not a complete solution in and of itself, but a useful tool in a larger security arsenal. Considering all this, we believe that TPMs could have a pivotal role in raising the bar for attackers.

References

[1] "Nykyisen toteutuksen haasteita — hackabi.org," https://hackabi.org/kilpailu/nykyisen-toteutuksen-haasteita.

[2] "TPM Main Specification — Trusted Computing Group," https://www.trustedcomputinggroup.org/tpm-main-specification.

[3] "Linux Integrity Subsystem," http://linux-ima.sourceforge.net.

[4] "Rohde-Schwarz-Cybersecurity/TrustedGRUB2: TPM enabled GRUB2 Bootloader," https://github.com/Rohde-Schwarz-Cybersecurity/TrustedGRUB2.

[5] "GNU GRUB - GNU Project - Free Software Foundation (FSF)," https://www.gnu.org/software/grub.

[6] "strongSwan - IPsec VPN for Linux, Android, FreeBSD, Mac OS X, Windows," https://www.strongswan.org.

[7] "TrustedNetworkConnect - strongSwan," https://wiki.strongswan.org/projects/strongswan/wiki/TrustedNetworkConnect.

[8] "Mutual Attestation of IoT Devices via Strongswan VPN," https://www.strongswan.org/docs/cebit2016.pdf.

[9] A. Steffen, "The linux integrity measurement architecture and tpm-based network endpoint assessment," Technical report, HSR University of Applied Sciences, Tech. Rep., 2012.

[10] "Minimum hardware requirements - Windows 10 hardware dev," https://msdn.microsoft.com/library/windows/hardware/dn915086(v=vs.85).aspx.

[11] "Trusted Platform Module (TPM) Summary — Trusted Computing Group," http://www.trustedcomputinggroup.org/trusted-platform-module-tpm-summary.