Exercise 3: Authenticated Boot

1 Introduction

1.1 Core Root of Trust for Measurement (CRTM)

The model of Trusted Computing defined by the TCG[1] defines Storage and Platform Integrity Management and Reporting to be one of the main features of Trusted Computing. This is used to build a mechanism called Authenticated Boot.

In the Trusted Computing model, the TPM[2] acts as the Root of Trust. As the TPM is a passive component, measurement will not be started by the TPM, but instead is the task of each bootstrapping component. The TCG defined the so-called Core Root of Trust for Measurement (CRTM). The CRTM is a piece of executable code starting the measurement of bootstrapping components. It is an immutable portion of the host platform's initialization code that executes upon platform reset. In contrast to the Dynamic Root of Trust for Measurement (DRTM) that was introduced with TCG's 1.2 specification, this fixed initialization code is sometimes also called Static Root of Trust for Measurement (SRTM).

It is important to note that trust in all subsequent measurements is based on the integrity of the CRTM. If the CRTM is tampered with, all subsequent measurements cannot be trusted. Ideally, the CRTM is located in the TPM, where it may be located in secure storage, but due to implementation decisions and the need to keep hardware costs of the TPM very low[3], the CRTM is mostly located in other firmware (e.g. BIOS boot block). It can be a part of the BIOS (e.g. BIOS is composed of a Boot Block and a POST BIOS) or the entire BIOS, where the latter choice would not be a good one, as this would increase the Trusted Computing Base.

1.2 Authenticated Boot

The overall flow of measurement and execution is depicted by figure 1.Upon system power-up, the TPM goes through a set of initialization and self-test functions. It then passes control to the CRTM which starts the chain of measurement by measuring and passing control to the BIOS. In general, every bootstrapping software S_i performs the following steps, where i is the position in the boot chain:

(1) Measure S_{i+1} by computing the hash[4] of the executable code of S_{i+1}
(2) Extend this measurement into the TPM's PCR[5] by using the TPM command $PCR\_Extend(n, S_{i+1}) : PCR_n = SHA1(PCR_n || S_{i+1}))$, where n is the PCR index and

[1]Trusted Computing Group
[2]Trusted Platform Module
[3]secure storage is expensive!
[4]for all TPM of version 1.2, the hash function is SHA-1
[5]Platform Configuration Register

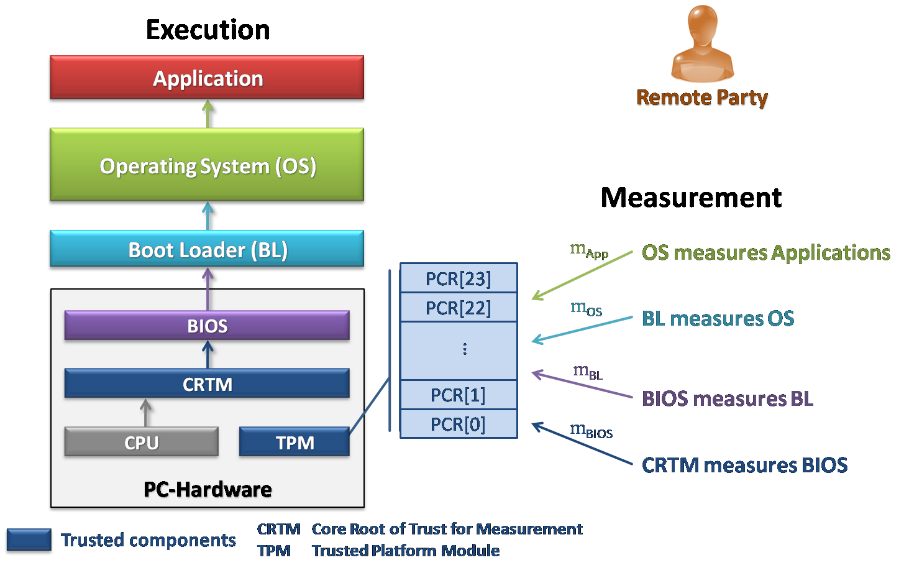

Figure 1: Authenticated Boot – Measurement and Execution Flow

(3) Finally pass control to S_{i+1}

Additionally, each time a PCR is extended, a log entry, e.g. containing name, version and vendor of the software, is made in the TCG Event Log. This allows a challenger to see how the final PCR digests were built.

The BIOS will perform the same steps before passing control to the bootloader. If the bootloader is TCG-enabled, like TrustedGRUB[5], it will continue by measuring important operating system files such as the kernel and kernel configuration files.

1.3 Authenticated vs. Secure Boot

After the operating system is finally booted, all measurements can be found in the corresponding Platform Configuration Registers. Usually you have a set of PCR values describing the initial, authentic state of the system. With *Authenticated Boot*, you can now detect any changes to the platform by comparing authentic to current PCR values. During *Secure Boot*, the authentic measurement values are compared to current measurements during the whole bootstrapping process and - if measurements differ - the boot process is cancelled and the system halted.

1.4 TrustedGRUB

TrustedGRUB[5] is an enhancement to the famous GNU GRUB[6] bootloader. It supports Authenticated Boot as well as Secure Boot using the Trusted Platform Module.

TrustedGRUB, in contrast to the original GRUB bootloader[6], consists of two parts - called stages. This architecture of multiple stages is historically related. In the beginnings of computer technology, the bootloader was designed to be placed at the first sector of the harddisk (head 0, track 0, sector 1) called Master Boot Record (MBR). The structure of the MBR is shown in figure 2. The MBR by convention is 512 bytes small. The bootloader must fit into the first 440 bytes of the MBR, followed by the partition table and additional disk signatures.

Address (Hex)	Address (Dec)	Description	Size in bytes
0000	0	Code Area (Bootloader)	440
01B8	440	Optional Disk Signature	4
01BC	444	Usually Nulls (0x0000)	2
01BE	446	Partition Table	64
01FE	510	MBR Signature 0xAA55	2

Figure 2: Structure of the Master Boot Record (MBR)

As the bootloaders added in more and more functionality like support for network boot, developers were compelled to find a way circumventing the size limit of only 440 byte. They split up the bootloader into several parts. A very small part, stage 1, is still located in the Master Boot Record. Upon boot, the CPU jumps to the MBR and executes stage 1. Stage 1 merely loads stage 2, which holds the filesystem drivers, program code for the selection screen and the GRUB shell as well as the load routine for the operating system kernel. Stage 2 would then load the operating system kernel and the bootstrapping process would continue.

Now for the whole bootloader, stage 1 and stage 2, to be measured by the BIOS, additional measurement functionality had to be added to stage 1 of GRUB. When executed, stage 1 would measure stage 2 before passing control to it. As the original stage 1 did nearly fit the whole MBR, code enabling network boot had to be removed in favor of the measurement code. As a side note, the SHA-1 measurements in TrustedGRUB are not done by the TPM via the $TCG\_HashAll()$ function, as tests revealed that the TPM is very slow in measuring code of bigger size. Instead, the bootloader will compute the hash itself and extend it into the PCRs. This will not break our security model but only minimally enlarges the Trusted Computing Base, because the bootloader is already a part of it.

1.4.1 Extending Authenticated Boot to the Operating System

Current TCG-enabled platforms feature a CRTM and a TCG-enabled BIOS. To extend the chain of measurements into the operating system, an enhanced bootloader that supports measurements is needed. TrustedGRUB is an enhanced bootloader that supports the measurement of the operating system kernel and arbitrary files on disk.

[6]original GRUB consists of an additional intermediate stage 1.5

1.4.2 Measurements

The measurements listed in figure 3 are done by the TrustedGRUB bootloader and other bootstrapping components. PCR_0 - PCR_7 are defined in TCG PC Client Specification[7], where all other PCRs can be used for additional measurements like, in our case, by TrustedGRUB.

PCR Index	PCR Usage
0	CRTM, BIOS and Platform Extensions
1	Platform Configuration
2	Option ROM Code
3	Option ROM Configuration and Data
4	IPL[7]Code (MBR Information and Bootloader Stage 1)
5	IPL Code and Configuration Data (for use by IPL Code)
6	State Transition and Wake Events
7	Reserved for future usage. Do not use.
8	Bootloader Stage 2 Part 1
9	Bootloader Stage 2 Part 2
10	Not in Use.
11	Not in Use.
12	Bootloader Commandline Arguments
13	Files checked via checkfile routine
14	Files which are actually loaded (e.g. Linux kernel, initrd, modules..)
15	Not in Use.
16	Not in Use.
17	DRTM[8]
18-23	Not in Use.

Figure 3: Summary of defined PCR Usage

1.4.3 Checkfile

TrustedGRUB measures important operating system files, such as the kernel to be booted, by default. Additionally, you can provide a list of files to be measured upon boot together with their authentic hash value. At the end of stage 2, TrustedGRUB will measure the files, extend the measurements into PCR_{13} and compare the measurements to the hash values given in the checkfile. If they don't match, which means the files must have been modified, the user will be warned and can decide to continue booting the system or not (Secure Boot).
The syntax of the checkfile is as follows:

[7]Initial Program Loader, e.g. Bootloader
[8]Dynamic Root of Trust for Measurement

HASH (hdX,Y)/some/path/some.file

Pay special attention that the drive parameter and the path are both correct, otherwise the system is not able to boot. The checkfile contains a list of entries itself that must not be be larger than 8096 bytes altogether. The path of each entry (including the last one) must be succeeded by a newline ('\n'). In the last place, a checkfile entry must be added to each GRUB *menu.lst*-entry, giving the filesystem path of the checkfile. For an example entry, see figure 4.

```
title          Ubuntu 8.04.1, kernel 2.6.24-21-generic
root           (hd0,4)
kernel         /boot/vmlinuz-2.6.24-21-generic root=...
initrd         /boot/initrd.img-2.6.24-21-generic
checkfile      /boot/grub/checkfile
quiet
```

Figure 4: GRUB menu.lst example entry with checkfile option added

2 Theoretical Assignments (6 Points)

1. What is the CRTM? Which bootstrapping component can you still trust when the CRTM is tampered?

2. Where should the CRTM be located? How could one update the CRTM but also protect it from being tampered this way?

3. Should the TCG Event Log be protected, e.g. it's integrity? Justify your answer!

4. TrustedGRUB measures important operating system files, such as the kernel, during the bootstrapping process. Can a kernel, such as the Linux kernel, change during runtime and which effects does this have? Give some examples!

5. The checkfile itself is not measured in any way. Think of the consequences! What can you do to protect the checkfile?

3 Practical Assignments

3.1 Using TrustedGRUB (3 Points)

1. TrustedGRUB is already installed on your system. It operates in "SHA-1 Mode", showing the measured SHA-1 result for every measured file loaded.

2. You can find all the TrustedGRUB files in the */boot/grub/* directory. List the contents of the TrustedGRUB directory and figure out their function.

3. The *menu.lst* file holds a list of all operating systems managed by TrustedGRUB. List the operating systems managed by TrustedGRUB on your platform and note on which partition they have been installed. Use the GRUB notation *(hdX,Y)*!

3.2 Measurements (3 Points)

In the following task, we let TrustedGRUB perform the measurements of important operating system files and validate the PCR value using a command line tool afterwards.

1. TrustedGRUB performs measurements of important operating system files being loaded. This includes the kernel and the initrd-image[9][10]. These measurements all are extended into PCR_{14}.

2. Check the measurements using the TrustedGRUB tool `verify_pcr`. Note both the PCR14 value and the output of `verify_pcr`.

[9] /boot/vmlinuz-XY, /boot/initrd.img-XY (XY: kernel version)

[10] initial ramdisk; temporary filesystem loaded by the kernel during boot, contains files needed for startup of the system

Usage: `verify_pcr <pcr initial value {NULL |20 byte hex}>{filename-1 ... filename-n}`

3. When measuring and extending more than one file into a PCR register, the order is very important. Explain why with regards to the order of the chain of trust!

3.3 Using the Checkfile (8 Points)

In the following task we add two files to the TrustedGRUB *checkfile*, let TrustedGRUB perform measurements of these additional files and validate the corresponding PCR value using a command line tool afterwards.

1. Add the system files */etc/passwd* and */etc/shadow* to the checkfile. Use the syntax mentioned in the introduction and don't forget to add a new line after each entry (including the last one)!
2. Compute the initial SHA-1 hash of both of these files and add it to the corresponding entry in the checkfile. You can either use the `create_sha1` tool of TrustedGRUB or the `openssl` package (see `man openssl`). Note both entries!
3. Now, reboot the system.
4. In which Platform Configuration Register (PCR) can you find the measurements of the files in your checkfile?

5. Now, verify the measurement of */etc/passwd* and */etc/shadow*. Use the *verify_pcr* tool provided by TrustedGRUB and compare the output with the corresponding PCR value. Note both values!

6. Make a change to either of these files, e.g. set a new password for user "labor" by executing `passwd labor`. Now, reboot again. What happens? Again, verify the PCR and note both values.

7. Suppose you are an attacker that wants to tamper the system. Your previous modification of the */etc/passwd* respectively */etc/shadow* file could be such a scenario.
   a) Are your system modifications detected? Justify your answer!
   b) How could you hide your modifications, so they cannot be detected?

Appendix

`create_sha1` - takes as an argument a filename and writes the result of the SHA-1 function of the file to *stdout*

`verify_pcr` - intended to check, if a PCR register is extended correctly with the given files. For example, if you have a checkfile containing 5 entries, all the files are hashed and extended into PCR_{13}. The corresponding value of PCR_{13} can be verified with this utility. Execute the command with the following parameters:

Usage: `verify_pcr <pcr initial value {NULL |20 byte hex}>{filename-1 ... filename-n}`

Example: All files which are actually loaded are hashed and extended into PCR_{14}. If you boot your Linux, this PCR would contain the Linux kernel (and probably initrd-files).

References

[1] Official Website of Trusted Computing Group, https://www.trustedcomputinggroup.org/

[2] Official Website of Lecture "Trusted Computing", http://www.ei.rub.de/studierende/lehrveranstaltungen/231/

[3] Trusted Computing Group Specification, https://www.trustedcomputinggroup.org/specs/

[4] TrouSerS Project Page, http://trousers.sourceforge.net/

[5] TrustedGRUB Project Page, http://www.sirrix.de/content/pages/50553.htm

[6] GNU GRUB Website, http://www.gnu.org/software/grub/

[7] TCG PC Client Specification, https://www.trustedcomputinggroup.org/specs/PCClient