Lab: Trusted Computing

Execution in a system controlled by the adversary

Daniel Bosk

Department of Information and Communication Systems
Mid Sweden University, SE-851 70 Sundsvall

1 Introduction

This laboratory assignment treats the area of trusted computing and the problem facing Digital Rights Management (DRM). The problem with DRM systems is that they must protect something in a hostile environment where the adversary controls everything.

A possible solution to this problem is to introduce a hardware module as support. One such module is the Trusted Platform Module (TPM), which was introduced in a cooperation between Microsoft, Intel, IBM, HP, and Compaq [1]. Their purpose for that was to support DRM. A newer development in this area is the UEFI secure boot, functionality utilized by the latest versions of Microsoft Windows, where the hardware refuses to boot the operating system if it is not cryptographically signed by a given key [4]. This way they can guarantee that the boot loader is not modified, the boot loader can further verify the rest of the operating system. Since the operating system is guaranteed to be unmodified, correct operation from the operating system can be expected.

However, without this support we will see that it is basically impossible to protect programs from modification or data from copying. The only reason the TPM prevents this is because it is hard for the adversary to modify this hardware.

1.1 Aim

The main aim of this assignment is that you should reflect on the possibilities of the adversary with unlimited access to the software running on a machine. More specifically, the expected learning outcomes of this assignment are that you will:

- Have insight into the problems concerning digital right management systems.
- Be able to reflect on the security of software with a trusted computing base with hardware support, e.g. TPM.

The next section covers what you must read before you understand this assignment and how to do the work. Section 3 covers the work to be done, i.e. how you should learn this. Section 4 covers how it will be examined, i.e. how you show that you have fulfilled the intended learning outcomes given above.

2 Theory

For this assignment you should first read Chap. 3–5, 16, 18, and 22 in *Security Engineering* [1]. Then you should read Chap. 10, 14–15 in *Computer Security* [3].

After reading the material given above you need to know about programming in assembler, specifically x86-64 assembler and some tools. For this you should read *x86-64 Machine-Level Programming* by Bryant [2]. You also need to be acquainted with some tools, study the manual pages for objdump(1), as(1), and gdb(1).

3 Assignment

There is a program with a very simple DRM found in URL

`https://github.com/OpenSecEd/drmlab/releases/download/v1.0/cpager-drm`.

It is an ELF 64-bit LSB executable (x86-64, dynamically linked, stripped) for a GNU/Linux system. You can find the instruction for how to use the program (without DRM) in Sect. 4.1 (specifically List. 1 and 2) of the document at URL:

`https://github.com/dbosk/opsys/releases/download/v1.0/labs-paging.pdf`.

The first part of this assignment is to break that DRM. This will be solved together during a full-class hackathon in the computer lab. There will be a projector with the code for all to see, then we will rotate who will be by the keyboard writing what the rest of the class is saying. This way we will discuss together and write the code together, everyone will thus participate in the process.

The second part of the assignment is to discuss the consequences of this, among other things we will discuss the following questions:

- What is the purpose of DRMs, since they can be circumvented?
- Can we implement a DRM which actually works? What do we need for this? Would it be worth it?

4 Examination

To pass this assignment you must first *actively* participate in the hackathon lab session. You must also actively contribute to the post-coding discussions. If you cannot participate in the lab session you have to solve the lab yourself, then orally present your solution during one of the lab sessions.

Acknowledgements This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported license. To view a copy of this license, visit `http://creativecommons.org/licenses/by-sa/3.0/`. Its original source code can be found in URL `https://github.com/dbosk/drmlab/`.

References

[1] Ross J. Anderson. *Security Engineering. A guide to building dependable distributed systems.* 2nd ed. Indianapolis, IN: Wiley, 2008. ISBN: 978-0-470-06852-6 (hbk.) URL: `http://www.cl.cam.ac.uk/~rja14/book.html`.

[2] David R. Bryant Randal E. and O'Hallaron. *x86-64 Machine-Level Programming.* Sept. 2005. URL: `https://www.cs.cmu.edu/~fp/courses/15213-s07/misc/asm64-handout.pdf`.

[3] Dieter Gollmann. *Computer Security.* 3rd ed. Chichester, West Sussex, U.K.: Wiley, 2011. ISBN: 9780470741153 (pbk.)

[4] Microsoft. *Secure Boot Overview.* Feb. 2014. URL: `http://technet.microsoft.com/en-us/library/hh824987.aspx`.