TPM Tutorial Worksheet

Ian Oliver
Crim 2024
November 13th 2024
University of Oulu

Contents

1 Installation

Firstly take note of these instructions:

- **READ THE INSTRUCTIONS CAREFULLY**.
- **WRITE NOTES AS YOU GO ALONG**.
- **READ THE INSTRUCTIONS CAREFULLY**.

The course is found at `https://github.com/iolivergithub/TPMCourse`. The README file contains the installation instructions carefully and make sure that you have this working before the tutorial.

The docker based installation has been tested in Linux (various Ubuntus and Debians).

If you are running on Windows, Mac or something even more exotic (you mean you don't an IBM zSeries at home?!) then it is probably much easier to build a virtual machine using VMWare Player or VirtualBox. Download Ubuntu desktop or server as you wish (Desktop has a nice UI!) and install and run it in your VM.

For those of you who can unpick the Dockerfile and figure out how to install everything on a bare-metal Linux box with a real TPM...good luck. If you brick your TPM, corrupt your OS, lose encryption keys or anything else, I did warn you and take no responsibility.

Once you get it working proceed to the exercises in section 2 of this document - it is immediately after this.

2 Tutorial and TPM Familiarity

Read each section carefully and take notes as you go along. If you break your simulated TPM, exit docker and restart. Note, if you had keys or data in that simulated TPM it will be lost on each startup.

Good Advice: keep a log of each command you run in a separate textfile - easy to copy and paste these back in later and also makes debugging easier. Finally **READ THE INSTRUCTIONS CAREFULLY**.

2.1 Basic Commands

Follow the tutorial commands outlined in the tutorial in this order. You will find these files under the 'docs' director of the downloaded course or directly on the Github pages at `https://github.com/nokia/TPMCourse/blob/master/docs/STARTHERE.md`

Take note of the warning in section 2.2 below.

Take notes as you go along, **draw pictures**, keep a log of commands and don't be afraid to restart the simulator if you need and **read the instructions carefully**.

- Random
- Objects
- Keys (see section 2.3 below)

- NVRAM
- PCRs
- Quoting

2.2 Command Parameters

Sometimes parameters to commands change - we use the latest TPM2 Tools and sometimes things do change and the course might not be fully updated. In this case, either make an issue in github or make a fork, change and then a pull request - your contributions either way will be very much appreciated.

2.3 TPM Memory and Flush Context

This can also be found in the section on key generation, but I will repeat it here. See also `https://github.com/iolivergithub/TPMCourse/blob/master/docs/keys.md#tpm-memory`

TPMs have limited space for storing objects such as keys, session information etc. Different manufacturers provide different amounts of space and in some cases even behaviours when dealing with temporary objects. The IBM TPM Simulator used in the Dockerfiles has very limited storage and you'll find objects being left in the transient memory space.

If you receive out of memory errors from the simulator or any TPM then check if there are objects (typically keys) in the transient memory area; for example we can use `tpm2_getcap` to display this and in this example we have two objects in the transient area.

```
$tpm2_getcap handles-transient
0x80000000
0x80000001
```

To remove these objects use the command `tpm2_flushcontext -t` and check with `tpm2_getcap` again - if nothing is reported then all worked.

```
$tpm2_getcap handles-transient
0x80000000
0x80000001
$tpm2_flushcontext -t
$tpm2_getcap handles-transient
```

2.4 Communicating with the TPM

If you are running against a real TPM, make sure you have access to the device. For example on Linux this means access to /dev/tpm0 and /dev/tpmrm0. In most distributions, and after installing the TPM2 Tools you will need to add your user to the tss group to get access to /dev/tpm0.

If you are running against the TPM simulator, then it is likely you will need to set an environment variable to direct commands to the simulator. Set this `export TPM2TOOLS_TCTI="mssim:host=localhost`

You can run the simulator on a machine with a physical TPM too, just set the environment variable above to direct tpm2* commands to the correct place.

3 Exercises

Follow each of these in turn. Refer back to the tutorial section and your notes.

3.1 Generate Keys

Generate an RSA key under one of the hierarchies. I suggest creating a primary key in owner first and then deriving from that.

Using this key encrypt a small amount of data, eg: 16 characters maximum.

Decrypt that data and demonstrate that that the encrypt/decrypt works.

3.2 Generate and EK and AK

The EK is the device's identity in a way. Generate this.

Then geneate an attestation key (AK).

Make both of these persistent at suitable handles.

3.3 PCRs

List the PCRs and use *tpm2_pcrevent* to extend certain some PCRs. Pick some small files to hash with this command - hashing takes time and the TPM isn't a cryptoaccelerator!

If you can reset the PCRs (something you can't do without a reboot in reality!), reset them and extend the PCRs using the same data but in different order. Compare what you get.

3.4 UEFI EventLog

You must be running on a Linux machine with UEFI enabled

Read the UEFI eventlog which can be found at */sys/kernel/security/tpm0* using *tmp2_eventlog*. You will need to use sudo and write the output of the command to your home directory or somewhere suitable. Give this file a suitable name - you will need it later. Read it with a text editor.

- What does the event log tell you?
- How does it relate to the PCRs?

Reboot your machine and enter the UEFI setup page. If you can find something *SAFE* to change, do it. If you are unsure, don't touch anything. Save and Exit and let the machine reboot.

- Read the event log again as above
- Compare it with the early event log you saved (you did this, yes?)
- What has changed

Did any PCRs? change - see section 3.3.

3.5 Quotes

Take two quotes of your machine for the same PCRs. Use *tpm2_print* them. How do their differ?

Suspend your machine, wake it up and take a quote, what has changed?

Reboot your machine and take a quote, what has changed?

NB: this following exercise is dangeous and should not be attempted - disk corruption, breaking hardware etc. You have been warned

Remove the powersupply and pull the battery from your machine if you can. Reboot, take a quote and what does it tell you?

3.6 Updates

Linux has various ways of updating itself, eg: *apt update/upgade, yum etc...* Take quotes, copies of the event log and PCRs *BEFORE* the update. Reboot, and take quotes, eventlog and PCRs again. What has changed.

Linux has a firmware update mechanism called LVFS and is used via the command *fwupdmgr*. Take quotes, copies of the event log and PCRs *BEFORE* the update. Reboot, and take quotes, eventlog and PCRs again. What has changed?

4 Challenges

Here are two challenges, the first is easy and the latter is harder - respectively. You should attempt both.

4.1 Encrypted Communication

Make yourself familiar with the idea of asymmetric encryption and utilising private and public keys to encrypt messages.

Start by creating a public/private key pair on your TPM. Distribute the public part to some else to load on their TPM. Encrypt a message utilsing your private key and send it to the recipient to decrypt using your supplied public key.

Now do the same by swapping public keys and sending your message encrypted using both keys so the recipient knows the message came from you and only they can decrypt.

4.2 Remote Attestation

This is HARD and requires installation of a remote attestation server. This task will take a number of hours and for best results requires machines (x86 PCs, Raspberry Pis) with physical TPMs. This will only work on Linux machines.

The software can be found here: `https://github.com/iolivergithub/jane`. Docker installations can also be found using the supplied instructions.

You will need to install Tarzan - the trust agent - on the machine with a TPM, create a record for it using Jane and then using the attest and verification functionality to monitor the machine.