TPM Laboratory I

Marcel Selhorst
m.selhorst@sirrix.com
What's this?

00 C4 00 00 01 3A 00 00 00 00 00 00 00 01 00 03 00 01 00
00 00 0C 00 00 08 00 00 00 00 02 00 00 00 00 00 00 00 01 00
DC FC C6 46 3A 97 F0 D9 F2 AB AA 90 82 C6 CC 09 00 50 3F
76 8E FD 07 03 02 0E 6F 08 D1 5E 47 38 2C 20 86 B1 62 1F
4A 81 08 1B 54 83 BD 21 E8 45 4F 58 60 50 CF 5F 88 15 07
0B E1 6C A0 A4 50 5A 53 08 33 A6 D0 B4 05 0B 0B AD 69 36
1E 24 10 91 ED DE A0 BC 97 5B D5 7E A2 BD DA 0F B6 6C D4
53 6F 77 18 4F 2C B6 36 8D 31 89 B3 92 76 69 DF 58 5D 13
2F 09 53 58 A2 57 B7 63 25 D2 F1 9B 9D E5 65 EB 73 70 CE
FF 79 0D 89 86 B7 DB 4D 5A 50 AC AC 4E 3C 86 80 8E C0 D0
81 EA 60 5E BB A4 37 B7 E1 AB 79 46 A0 E4 03 CD 69 40 94
13 84 5C 6A A6 A6 09 D9 1B 3D 90 4E 66 5D 5B E6 53 4E 57
92 32 42 2C 45 37 F5 FC 19 7B 7D 45 49 07 F8 51 56 97 57
5D 9B EC F7 8C 14 A6 AF BF 0B B9 7D D8 89 62 65 45 89 99
A8 67 C9 37 47 49 E8 A6 DA 66 F5 00 FD ED 6D 43 69 94 AD
33 C8 B3 E6 16 86 38 14 DB 23 BA 2E E8 E4 32 1D FF BE 88
E4 76 6C 1C C9 5E C0 E3 C9 64
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
- TPM commands
- The Chain of Trust
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
- TPM commands
- The Chain of Trust
Introduction to TPMs (1)

- Hardware-based random number generators
- Cryptographic functions
  - Signatures, Hash (SHA-1), Encryption (RSA), Key generation
- Platform Configuration Registers (PCR)
  - Storage of integrity measurements
Introduction to TPMs (2)

- TPMs main goals
  - Security Anchor inside the system
  - Sealing / Binding to a certain Platform configuration
  - Attestation of the platform state
- Every TPM has a unique key called **Endorsement Key**
- The TPM has 8 states:

<table>
<thead>
<tr>
<th>Modes of Operation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 – Enabled – Active – Owned</td>
</tr>
<tr>
<td>S2 – Disabled – Active – Owned</td>
</tr>
<tr>
<td>S3 – Enabled – Inactive – Owned</td>
</tr>
<tr>
<td>S4 – Disabled – Inactive – Owned</td>
</tr>
<tr>
<td>S5 – Enabled – Active – Unowned</td>
</tr>
<tr>
<td>S6 – Disabled – Active – Unowned</td>
</tr>
<tr>
<td>S7 – Enabled – Inactive – Unowned</td>
</tr>
<tr>
<td>S8 – Disabled – Inactive – Unowned</td>
</tr>
</tbody>
</table>
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
- TPM commands
- The Chain of Trust
Platform Integration (1)

- TPMs are disabled by default
Platform Integration (2)

- Enable the TPM in the BIOS
  - Press F10 to enter BIOS
  - Enter “tpm” as Setup Password
Platform Integration (3)

- Enable the TPM in the BIOS
  - Security -> TPM Embedded Security
  - Embedded Security Device State -> „Enable“
Platform Integration (4)

- Enable the TPM in the BIOS
  - „Save Changes And Exit”
Platform Integration (5)

- Enable the TPM in the BIOS
  - Press “F1” to Accept
Platform Integration (6)

- Now the TPM should be available and detected!
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
  - TPM device drivers
  - TPM open source software
    - TrouSerS
    - Taking Ownership with TPM-Manager
- TPM commands
- The Chain of Trust
TPM device drivers (1)

- In order to use a TPM with Linux, a TPM device driver has to be available
- Currently, the following device drivers are available within any modern Linux kernel:
  - Atmel TPM 1.1b
    - `modprobe tpm_atmel`
  - NSC TPM 1.1b
    - `modprobe tpm_nsc`
  - Infineon TPM 1.1b + TPM 1.2
    - `modprobe tpm_infineon`
  - Generic TIS driver for TPMs 1.2
    - `modprobe tpm_tis`
TPM device drivers (2)
TPM device drivers (3)
TPM device drivers (4)
TPM device drivers (5)
TPM device drivers (6)
TPM device drivers (7)
TPM device drivers (8)

- HP compaq 6715b laptops we are using are equipped with an Infineon TPM 1.2
- Therefore, we have two options on TPM-device drivers:
  - legacy Infineon TPM 1.2
    - `modprobe tpm_infineon`
  - Generic TIS driver for TPMs 1.2
    - `modprobe tpm_tis`
- We will use `tpm_infineon`
TPM device drivers (9)

- Step 1: Open a Terminal / Konsole
  - click on this desktop icon
- Step 2: Load the device driver
  - `sudo modprobe tpm_infineon`
- Step 3: Verify, that the TPM device driver has loaded successfully:
  - `lsmod | grep tpm`
  - `dmesg | grep tpm`
- Step 4: Verify the existence of the correct node-device
  - `ls -l /dev/tpm0`
    - should be user:tss, 10, 224
TPM device drivers (10)

- Step 5: Verify the existence of *sysfs*-directory
  - `ls -l /sys/class/misc/tpm0/device`
    - caps
    - id
    - options
    - pcrs
    - pubek
    - resources
    - ...

- Now we should be able to communicate with the TPM via `/dev/tpm0`
TPM device drivers (11)

- Read out the Platform Configuration Registers (PCRs)
  - `cat /sys/class/misc/tpm0/device/pcrs`
  
  ![Screenshot of command output]

- We will come back to PCRs in section “Chain of Trust”
TPM device drivers (12)

- Read out the Public Endorsement Key
  - `cat /sys/class/misc/tpm0/device/pubek`

Now we should be able to communicate with the TPM via `/dev/tpm0`
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
  - TPM device drivers
  - TPM open source software
    - TrouSerS
    - Taking Ownership with TPM-Manager
- TPM commands
- The Chain of Trust
TPM open source software (1)

- Available Open Source Software:
  - TrustedGRUB
    - http://sourceforge.net/projects/trustedgrub
  - TrouSerS TSS
    - http://sourceforge.net/projects/trousers
  - TPM-Tools
    - http://sourceforge.net/projects/trousers
  - OpenSSL TPM-Engine
    - http://sourceforge.net/projects/trousers
  - TPM-Manager
    - http://sourceforge.net/projects/tpmmanager
TPM open source software (2)

- Open Source Software high-level hierarchy

- TPM Manager
- TrouSerS TPM Tools
- TrouSerS OpenSSL TPM Engine
- TrouSerS Trusted Software Stack (TSS)
- Linux Trusted Device Driver (TDD)
- TrustedGRUB Boot Loader

→ Chap. 5
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
  - TPM device drivers
  - TPM open source software
    - TrouSerS
    - Taking Ownership with TPM-Manager
- TPM commands
- The Chain of Trust
TPM open source software (3)

- TrouSerS TSS architecture
TPM open source software (4)

- TrouSerS TPM communication
TPM open source software (5)

- Start TrouSerS:

  ~$ sudo /etc/init.d/tcsd start

  * Starting TrouSerS' TCS daemon (tcsd) ... [ ok ]

- Start the TPM-Manager:

  ~$ tpmmanager
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
  - TPM device drivers
  - TPM open source software
    - TrouSerS
    - Taking Ownership with TPM-Manager
- TPM commands
- The Chain of Trust
TPM open source software (6)

- TPM-Manager:
TPM open source software (7)

- TPM-Manager:
TPM open source software (8)

- **TPM-Manager:**
TPM open source software (9)

- Taking Ownership with the TPM-Manager:
TPM open source software (10)

- Taking Ownership with the TPM-Manager:

- Set SRK Password
  - You can choose the WELL_KNOWN_SECRET for SRK as default value. You may be also want to set it manually.

- Set SRK to WELL_KNOWN_PASSWORD
  - Set SRK manually

- Take Ownership: Set Owner Password
  - Enter the TPM owner password. The owner has the right to perform special operations on the TPM.

  - Password: *********
  - Verify: *********
  - Password strength meter: 
  - Passwords match

Help  [OK]  Cancel
TPM open source software (11)

- Taking Ownership with the TPM-Manager:
TPM open source software (12)

- Taking Ownership – TSS interaction:

1. slotTakeOwnership
   1.1: readOwnerPassword
   1.2: Owner Password
   1.3: readSRKPassword
   1.4: SRK Password

1.5: takeOwnership

1.5.1: GetPolicyObject
   1.5.3: Policy_SetSecret
   1.5.5: Context_CreateObject
   1.5.7: GetPolicyObject
   1.5.9: Policy_SetSecret
   1.5.11: TPM_TakeOwnership

1.7: Result Dialog
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
- TPM commands
- The Chain of Trust
TPM commands (1)

- The current TCG specification 1.2 rev. 103 has > 100 TPM commands
- TPM commands are classified into 5 categories
  - Mandatory
  - Optional
  - Deprecated
  - Deleted
  - Vendor-specific
- The TCG spec. part 3 defines all input and output parameters for the available commands
- The TCG spec. part 2 defines the actual values for the parameters, structures, commands etc.
TPM commands (2)

- Every command consists of at least 3 parameters:
  - **TPM_TAG** `tag`  
    -> defines the degree of authorization
  - **UINT32** `paramSize`  
    -> defines the total amount of input bytes
  - **TPM_COMMAND_CODE** `ordinal`  
    -> represents the function, the TPM shall execute

- Every TPM command will be processed by the TPM
- The TPM always responds with a return code
TPM commands (3)

- There are 1 + 4 types of return codes
- Successful operation:
  - Return code is TPM_SUCCESS (0x0)
  - Additional data might be included in the response (e.g., if a key shall be created, the response will also contain the generated key)
- Error during operation:
  - When a command fails for any reason, the TPM must return only the following 3 items:
    - TPM_TAG_RQU_COMMAND (2 bytes)
    - ParamLength(4 bytes, fixed at 10)
    - Return Code (4 bytes, never TPM_SUCCESS)
TPM commands (4)

- The return codes for errors are divided into 4 categories:
  - TPM defined fatal errors
    - (0x001 to 0x3FF)
  - Vendor defined fatal errors
    - (0x400 to 0x7FF)
  - TPM defined non-fatal errors
    - (0x800 to 0xBFF)
  - Vendor defined non-fatal errors
    - (0xC00 to 0xFFF)
TPM commands (5)

Currently 99 TPM defined fatal errors:
  • Defined in TPM spec. part 2, pages 131ff

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPM_AUTH_FAIL</td>
<td>TPM_BASE +1</td>
<td>Authentication failed</td>
</tr>
<tr>
<td>TPM_BADINDEX</td>
<td>TPM_BASE +2</td>
<td>The index to a PCR, DIR or other register is incorrect</td>
</tr>
<tr>
<td>TPM_BAD_PARAMETER</td>
<td>TPM_BASE +3</td>
<td>One or more parameter is bad</td>
</tr>
<tr>
<td>TPM_AUDIT_FAILURE</td>
<td>TPM_BASE +4</td>
<td>An operation completed successfully but the auditing of that operation failed</td>
</tr>
<tr>
<td>TPM_CLEAR_DISABLED</td>
<td>TPM_BASE +5</td>
<td>The clear disable flag is set and all clear operations now require physical access</td>
</tr>
<tr>
<td>TPM_DEACTIVATED</td>
<td>TPM_BASE +6</td>
<td>The TPM is deactivated</td>
</tr>
<tr>
<td>TPM_DISABLED</td>
<td>TPM_BASE +7</td>
<td>The TPM is disabled</td>
</tr>
<tr>
<td>TPM_DISABLED_CMD</td>
<td>TPM_BASE +8</td>
<td>The target command has been disabled</td>
</tr>
<tr>
<td>TPM_FAIL</td>
<td>TPM_BASE +9</td>
<td>The operation failed</td>
</tr>
<tr>
<td>TPM_BAD_ORDINAL</td>
<td>TPM_BASE +10</td>
<td>The ordinal was unknown or inconsistent</td>
</tr>
<tr>
<td>TPM_INSTALL_DISABLED</td>
<td>TPM_BASE +11</td>
<td>The ability to install an owner is disabled</td>
</tr>
<tr>
<td>TPM_INVALID_KEYHANDLE</td>
<td>TPM_BASE +12</td>
<td>The key handle can not be interpreted</td>
</tr>
<tr>
<td>TPM_KEYNOTFOUND</td>
<td>TPM_BASE +13</td>
<td>The key handle points to an invalid key</td>
</tr>
</tbody>
</table>
TPM commands (6)

- Example of the input / output parameters of the TPM command `TPM_PCRRead`

<table>
<thead>
<tr>
<th>PARAM</th>
<th>HMAC</th>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>#</td>
<td><code>TPM_TAG</code></td>
<td><code>tag</code></td>
<td><code>TPM_TAG_RQU_COMMAND</code></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td><code>UINT32</code></td>
<td><code>paramSize</code></td>
<td>Total number of input bytes including <code>paramSize</code> and <code>tag</code></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td><code>TPM_COMMAND_CODE</code></td>
<td><code>ordinal</code></td>
<td>Command ordinal: <code>TPM_ORD_PCRRead</code></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td><code>TPM_PCRINDEX</code></td>
<td><code>pcrIndex</code></td>
<td>Index of the PCR to be read</td>
</tr>
</tbody>
</table>

**Outgoing Operands and Sizes**

<table>
<thead>
<tr>
<th>PARAM</th>
<th>HMAC</th>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>#</td>
<td><code>TPM_TAG</code></td>
<td><code>tag</code></td>
<td><code>TPM_TAG_RSP_COMMAND</code></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td><code>UINT32</code></td>
<td><code>paramSize</code></td>
<td>Total number of output bytes including <code>paramSize</code> and <code>tag</code></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td><code>TPM_RESULT</code></td>
<td><code>returnCode</code></td>
<td>The return code of the operation.</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td><code>TPM_COMMAND_CODE</code></td>
<td><code>ordinal</code></td>
<td>Command ordinal: <code>TPM_ORD_PCRRead</code></td>
</tr>
<tr>
<td>4</td>
<td>3S</td>
<td><code>TPM_PCRVALUE</code></td>
<td><code>outDigest</code></td>
<td>The current contents of the named PCR</td>
</tr>
</tbody>
</table>
TPM commands – exercises

- Now we are going to write some TPM commands
- The needed TCG specification is located at:
  /home/etiss/Desktop/exercise/spec
- There are two different versions available, since the latest revision 103 didn't update the table-of-content
- Before executing any of the commands, the TrouSerS-daemon `tcsd` has to be stopped!
  - `sudo /etc/init.d/tcsd stop`
- Since the permissions on `/dev/tpm0` belong to TrouSerS, we have to change this in order to gain read-write-access to the TPM device:
  - `sudo chmod 666 /dev/tpm0`
TPM commands – exercise 1

- /home/etiss/Desktop/exercise/code1/ contains a code skeleton using the TPM-command TPM_GetCapability. Please fill out the missing capability parameters:

  #define TPM_CAP_PROPERTY
  #define TPM_CAP_VERSION_VAL
  #define TPM_CAP_PROP_MANUFACTURER
  #define TPM_CAP_PROP_OWNER

  • make
  • ./tpm_getcapabilities
TPM commands – exercise 2

- /home/etiss/Desktop/exercise/code2/ contains a code skeleton of the TPM-command TPM_Extend. This code will hash a file and extend it into the defined PCR. Please fill out the missing command parameters:

```c
#define TPM_EXTEND_TAG
#define TPM_EXTEND_PARAMSIZE
#define TPM_EXTEND_ORDINAL
#define TPM_EXTEND_PCR_INDEX

• make
• ./extend_pcr <some filename>
```
Content

- Introduction to TPMs
- Platform Integration
- Using the TPM with Linux
- TPM commands
- The Chain of Trust
In the diagram, the execution flow is illustrated with a chain of trust. The trusted components include:

- Core Root of Trust for Measurement (CRTM)
- Trusted Platform Module (TPM)

**Hardware**:

- **CRTM**
- **TPM**

**Software**:

- **Bootloader (BL)**
- **Operating System (OS)**
- **App**

The diagram shows the following measurement relationships:

- **CRTM measures BIOS**
- **BL measures OS**
- **OS measures App**
- **BIOS TC measures BL**
- **CRTM measures BIOS**

A table with PCR values is shown:

<table>
<thead>
<tr>
<th>PCR Value</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCR[0]</td>
<td>BIOS</td>
</tr>
<tr>
<td>PCR[1]</td>
<td>BL</td>
</tr>
<tr>
<td>PCR[23]</td>
<td>OS</td>
</tr>
</tbody>
</table>

The instantiation is based on the TCG approach.
The Chain of Trust (2)

- New TCG BIOS commands added
  - Examples
    - TCG_StatusCheck: Checks whether a TPM is available
    - TCG_HashAll: Computes SHA1 hashes of given input data (boot loader)
    - TCG_PassThroughToTPM: Sends TPM commands to the TPM via BIOS TPM Driver
    - ...
  - Command calls via Interrupt 0x1Ah

- All commands have an Input Parameter Block (IPB) and an Output Parameter Block (OPB)
The Chain of Trust (3)
Detailed TCG BIOS Example

- TCG_PassThroughToTPM
- IPB: contains TPM command and parameters as specified in TCG specification (e.g., TPM_Extend)

On Entry:
- Ah: 0xBB  // TCG command
- Al: 0x02  // Function selector
  (here TCG_PassThroughToTPM)
- ES:DI:    // Pointer to IPB
- DS:SI:    // Pointer to OPB
- EBX: 0x41504354  // 'TCPA'
- ECX: 0
- EDX: 0
- Int: 0x1A  // Interrupt

On Return:
- EAX: TCG BIOS Return code
- DS:SI: Updated reference buffer (OPB)
The Chain of Trust (4) - TrustedGRUB

- According to TCG Specification measurements performed up to MBR
- TrustedGRUB extends the common available GRUB boot loader with mechanisms realizing authenticated boot up to OS
  - TrustedGRUB = TCG extended BIOS (CRTM) + GRUB + TPM functions
  - “stage1” measures subsequent stage “stage2”
  - “stage2” measures OS components (e.g., kernel), configuration file and optionally any additional files
- No direct communication with TPM
  - applies BIOS-calls instead (defined by the TCG)
The Chain of Trust (5) - TrustedGRUB

- Multiboot Modules
- OS Kernel
- Additional Files (optional)
- command list
- GRUB configuration
- Boot Loader Stage 2
- Boot Loader Stage 1
- BIOS
- CRTM
- CPU
- root host

Extended Boot Loader
Extended Hardware

is measured by transfers control to
The Chain of Trust (6) - TrustedGRUB

- **Platform Configuration Registers (PCRs)**

  ![Platform Configuration Registers](image)

  **Platform Configuration Register**
  - 00: BIOS
  - 01: Mainboard Configuration
  - 02: Option ROM
  - 03: Option ROM Configuration
  - 04: Initial Program Loader (IPL)
  - 05: IPL Config & Data
  - 06: RFU (Reserved for Future Usage)
  - 07: RFU
  - 08: First part of „stage2“
  - 09: Rest of „stage2“
  - 12: Commandline parameters
  - 13: Arbitrary file measurements (e.g., Kernel, modules,...)
  - 17-22: Resettable PCRs for DRTM
The Chain of Trust (7) - TrustedGRUB

- Platform Configuration Registers represent the current platform state
- The platform can only be trustworthy, if a complete, uninterrupted chain-of-trust exists
- PCRs can only be extended and are not resettable until platforms reboot (except for PCRs 17-22 in TPMs 1.2)
- By extending a PCR with a new measurement, the resulting value will be:

\[
PCR_{\text{new}} = \text{SHA1}(PCR_{\text{old}} || \text{newValue})
\]
The Chain of Trust (8) - TrustedGRUB

- To verify the content of PCRs 13 and 14 a tool exists called:
  
  `verify_pcr`

- Usage:
  
  `~ $ verify_pcr <initial PCR> <files 1-n>`

- Example:
  
  `~ $ verify_pcr NULL /boot/vmlinuz`
  
  Result for PCR:
  
  5c 16 04 82 0e 3e 52 68 71 b4
  8d 25 95 a5 8d 16 96 19 58 6d
Thank you!
Any Questions?

Marcel Selhorst
m.selhorst@sirrix.com