Federal Office
for Information Security

TPM Provisioning

Version: 1.0

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
Phone: +49 22899 9582-0
E-Mail: bsi@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

Figures

1 Introduction

This document implements the work plan outlined in Work Package 5B of the project “SiSyPHuS Win10: Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10“ (orig., ger.). The project is contracted by the German Federal Office for Information Security (orig., ger., Bundesamt für Sicherheit in der Informationstechnik - BSI). The work planned as part of Work Package 5B has been conducted by ERNW GmbH in the time period between August and September 2019, in accordance with the time plan agreed upon by ERNW GmbH and the German Federal Office for Information Security.

As required by the German Federal Office for Information Security, the version of the Trusted Platform Module (TPM) standard in focus is 2.0. The release of the Windows 10 system in focus is Windows 10 Enterprise long-term servicing branch (LTSB) 2019. This work package is an extension of Work Package 5, conducted by ERNW GmbH in the time period between July and September 2017.

The objective of this work package is the development of a proof-of-concept (PoC) for provisioning the TPM outside of the Windows operating system. The TPM provisioning process involves multiple activities, such as setting an owner authorization value and storing a storage root key (SRK) in the TPM device ((ERNW_WP5), Section 2.5). The focus of this work is on setting an owner authorization value ((ERNW_WP5), Section 1.3). This process involves changing an already set owner authorization value, or specifying a new owner authorization value, and storing this value in the TPM. In this work, we use the `tpm2-tools` Linux toolset of version 3.1.3-2 (git_tpm2tools, 2019) for setting an owner authorization value in Linux.

1.1 Concepts and terms

This section introduces concepts and terms relevant for understanding the contents of this work. The TPM provides a rich set of mechanisms for access control over resources managed by the TPM (i.e. objects). Objects can be managed by four entities (also referred to as management entities in this work): the platform firmware, the owner administrator, the endorsement administrator, and the lockout administrator. The access control mechanisms provided by the TPM enables the TPM to make a distinction between these entities when it comes to managing TPM objects. Each of the management entities has its own domain of objects for management. The following introduces relevant related terms:

Authorization values: Each management entity authorizes itself for performing management operations in its domain by using an authorization value. This value is used for authorizing the execution of specific protected TPM commands and functionalities. These commands and functionalities are protected such that they can be executed only if a proof of authorization is provided to the TPM. This proof is in the form of authorization values. Setting authorization values is the process referred to as taking ownership of the TPM. The TPM supports the following authorization values:

- platform authorization: The platform firmware uses the platform authorization value (`platformAuth`) for managing the objects in its domain. The platform firmware has access to all protected TPM commands and functionalities. To the domain managed by the platform firmware belongs the TPM hardware chipset.
- owner authorization: The owner administrator uses the owner authorization value (`ownerAuth`) for managing the objects in its domain. The owner administrator has access to some protected TPM commands and functionalities. To the domain managed by the owner administrator belongs the data and key storage mechanism of the TPM ((ERNW_WP5), Section 1.3).
- endorsement authorization: The endorsement administrator uses the endorsement authorization value (`endorsementAuth`) for managing the objects in its domain. The endorsement administrator has access to some protected TPM commands and functionalities. To the domain managed by the endorsement administrator belongs the platform identification mechanism of the TPM. This mechanism is relevant when the identity of a given TPM-enabled platform has to be proved.

- lockout authorization: The lockout administrator uses the lockout authorization value (`lockoutAuth`) for managing the objects in its domain. The lockout administrator has access to some protected TPM commands and functionalities. To the domain managed by the lockout administrator belongs the mechanism for protection against brute-force attacks, where an attacker tries different authorization values discussed in this section, until one succeeds. When the lockout mechanism has been triggered, the TPM prevents the usage of specific protected TPM commands and functionalities. In order to reset the lockout mechanism, the lockout administrator must provide the `lockoutAuth` value.

Authorization types: The TPM supports three authorization methods involving the authorization values discussed in this section: password, hash message authentication code (HMAC), and policy (tcgtpm, 2019). TPM users, such as the Windows 10 operating system, configure the authorization method to be used when taking ownership of the TPM. The password and HMAC authorization methods authorize TPM commands and functionalities based on knowledge of a secret stored in the TPM device. The policy authorization method, also known as extended authorization (EA), authorizes TPM commands and functionalities based on conditions. These conditions can be combined with the `AND` and `OR` logical operations (e.g., a specific Platform Configuration Register (PCR) state `AND` a correct secret).

Hierarchy: A hierarchy is collection of objects that are managed by a given management entity. At the root of a hierarchy is a root key to which other objects (e.g., keys and arbitrary data) may be attached. A hierarchy can be persistent (retained through a system reboot) or volatile (erased at each system reboot). The TPM provides three persistent hierarchies: the platform, endorsement, and owner hierarchy. These hierarchies belong to the domains managed by the platform firmware, the endorsement administrator, and the owner administrator, respectively.

The platform hierarchy enables secure storage for platform manufacturers to store platform-related objects. This includes cryptographic material to protect the update mechanisms of the core root of trust for measurement ((ERNW_WP5), Section 1.3) in order to comply with NIST SP 800-147 (see (nist_sp_800-147b, 2019)).

The endorsement hierarchy enables secure storage of objects relevant for proving the identity of a given TPM-enabled platform. This includes the endorsement key, which is unique for each manufactured TPM device.

The owner hierarchy enables secure storage of objects relevant to TPM users, including the Windows 10 operating system. This includes encryption of arbitrary data using the root key of the owner hierarchy (i.e., the SRK) and storage of encryption keys in the non-volatile memory of the TPM.

2 Technical Analysis

This section presents a proof-of-concept implementation for setting of an owner authorization value in Linux. In this work, we use the `tpm2-tools` toolset (git_tpm2tools, 2019) for provisioning the TPM in Linux, including setting an owner authorization value. Figure 1 depicts the use of the different tools that are part of `tpm2-tools` for provisioning a TPM device. The tool `tpm2_takeownership` is used for setting authorization values and storing them in the TPM, that is, for taking ownership of the TPM (see [1] in Figure 1). The `-o` parameter of `tpm2_takeownership` specifies the owner authorization value to be set. In the scenario depicted in Figure 1, the password authorization method is used. The tool `tpm2_createprimary` (see [2] in Figure 1) is used to generate an SRK under the set owner authorization value ((ERNW_WP5), Section 1.3). The tool `tpm2_evictcontrol` (see [3] in Figure 1) is used to store a previously generated SRK in a persistent, non-volatile, memory location in the TPM. The tools `tpm2_listpersistent` and `tpm2_readpublic` (see [4] in Figure 1) are used to view attributes of a previously generated SRK.

Figure 1: TPM provisioning with the tpm2-tools toolset

We now present how the owner authorization value, set as part of the TPM provisioning process depicted in Figure 1, can be changed with the `tpm2-tools` toolset across the boundaries of the operating system that has provisioned the TPM. To this end, we used the tool `tpm2_takeownership` to successfully change the previously set owner authorization value in another, live-booted Linux system (see Figure 2). In order to change a set owner authorization value, the `tpm2_takeownership` tool is executed such that:

- the `-O` parameter specifies the old owner authorization value; and
- the `-o` parameter specifies the new owner authorization value.

The output of the tools `tpm2_listpersistent` and `tpm2_readpublic` shows that changing the owner authorization value in the live-booted Linux system has no impact on the SRK generated as part of the TPM provisioning process conducted earlier. This shows that changing a set owner authorization value across the boundaries of Linux operating systems is technically feasible.

```
dphillips@0x0002:~|⇒ tpm2_takeownership -o "1111" -O "1234" -V
INFO on line: "114" in file: "tools/tpm2_takeownership.c": Successfully changed hierarchy for Owner
```

```
dphillips@0x0002:~|⇒ tpm2_listpersistent
persistent-handle[0]:0x81000001 key-alg:rsa hash-alg:sha256 object-attr:fixedtpm|fixedparent|sensitivedataorigin|userwithauth|noda|restricted|decrypt

dphillips@0x0002:~|⇒ tpm2_readpublic -H 0x81000001 -V
name: 000b276f90840f37ac43b3983676aca04d4bc0a02c6baeb6a126bbfdc1aaa73597fe
qualified name: 000be8fdeb5f1cf5968a002c3093b7525cf86372027e5df85a893776b3b298eb0b8e
algorithm:
  value: sha256
  raw: 0xb
attributes:
  value: fixedtpm|fixedparent|sensitivedataorigin|userwithauth|noda|restricted|decrypt
  raw: 0x30472
type:
  value: rsa
  raw: 0x1
  rsa: a258f849d95d5844290d2cf67f71027650719b9395a41aa831a3e859[...]d8a4cef9bffa1fb7807fcc98204af32a8c18a9d1f1dfe21026354a53
```

Figure 2: Changing a set owner authorization value with tpm2_takeownership

We now analyze whether changing a set owner authorization value across the boundaries of a Windows and a Linux operating system is technically feasible. We first analyze whether an owner authorization value, set during a TPM provisioning process conducted by Windows 10 (see (ERNW_WP5), Section 2.5), can be changed in a live-booted Linux system using the `tpm2_takeownership` tool. The owner authorization value generated by Windows 10 is a 20-byte long binary sequence. Windows 10 can be configured to store the owner authorization value in a Base64-encoded form at the registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TPM\WMI\Admin\OwnerAuthFull` (see (ERNW_WP5), Section 3.1.2). For this owner authorization value to be changed in Linux using the `tpm2_takeownership` tool, it has to be converted to its binary form so that it can be specified as the `-O` parameter of the tool. Figure 3 depicts a conversion of a Base64-encoded owner authorization value into its binary form using PowerShell commands (ERNW_WP8, n.d.).

```
PS C:\Users\dphillips> Get-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\TPM\WMI\Admin\
LastAuthLevel    : 4
OwnerAuthStatus  : 1
OwnerAuthFull    : jN4eiezBYiaytCBvf0eW4oxXZcg=
LockoutHash      : e031O2F9WZG4aiu5/iY6DkiLxqg=
[...]

PS C:\Users\dphillips> [System.Convert]::FromBase64String('jN4eiezBYiaytCBvf0eW4oxXZcg=').GetType()
IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Byte[]                                   System.Array

PS C:\Users\dphillips> ([System.BitConverter]::ToString([System.Convert]::FromBase64String('jN4eiezBYiaytCBvf0eW4oxXZcg='))).Replace("-","")
8CDE1E89ECC16226B2B4206F7F4796E28C5765C8
```

Figure 3: Conversion of a Base64-encoded owner authorization value into binary form

Figure 4 depicts the execution of the `tpm2_takeownership` tool for changing the owner authorization value that has been previously set during a TPM provisioning process conducted by Windows 10. The `-O` parameter of the tool specifies the set owner authorization value in its binary form (see Figure 3). The execution of `tpm2_takeownership` fails with the error code `0x9a2`. The `tpm2_rc_decode` tool, part of the `tpm2-tools` toolset, can be used to decode this code (see Figure 4). The output of `tpm2_rc_decode` indicates that there is an issue related with the owner authorization value specified by the `-O` parameter of `tpm2_takeownership`. We suspect that the issue is related to a second factor for authorizing the TPM functionality of changing an owner authorization value set in Windows 10. This may be the case in the scenario where Windows 10 configures the HMAC or policy authorization methods when taking ownership of the TPM. In order to evaluate this hypothesis, we attempted to change the lockout authorization value set during the TPM provisioning process conducted by Windows 10 (see (ERNW_WP5), Section 1.3). Windows 10 sets the same value both as an owner and as a lockout authorization value (see (ERNW_WP5), Section 2.5). Therefore, if changing the lockout authorization value succeeds, the error code `0x9a2` is not related to the owner authorization value specified by the `-O` parameter of `tpm2_takeownership` itself, but to a second factor for authorizing the TPM functionality of changing an owner authorization value set in Windows 10. Hence, the hypothesis is confirmed.

```
dphillips@0x0002:~|⇒ tpm2_takeownership -o "hex:9da86e6eadc1d07981e03e25ab1f429feacad511" -O "hex:8cde1e89ecc16226b2b4206f7f4796e28c5765c8" -V
ERROR on line: "109" in file: "tools/tpm2_takeownership.c": Could not change hierarchy for Owner. TPM Error:0x9a2
ERROR on line: "165" in file: "tools/tpm2_tool.c": Unable to run tpm2_takeownership

dphillips@0x0002:~|⇒ tpm2_rc_decode 0X9a2
error layer
  hex: 0x0
  identifier: TSS2_TPM_RC_LAYER
  description: Error produced by the TPM
format 1 error code
  hex: 0x22
  identifier: TPM2_RC_BAD_AUTH
  description: authorization failure without DA implications
session
  hex: 0x100
  identifier: TPM2_RC_1
  description:  (null)
```

Figure 4: Changing an owner authorization value set in Windows 10

We attempted to change the set lockout authorization value with the `tpm2_takeownership` tool, such that the `-L` parameter specifies the set lockout authorization value (i.e., the set owner authorization value, see Figure 5). This attempt succeeds (see `Successfully changed hierarchy for Lockout` in Figure 5). This indicates that changing an owner authorization value set in Windows 10 is a TPM functionality that is protected by a second authorization factor, in addition to the owner authorization value itself. This, in turn, indicates that changing an owner authorization value set in a given Windows 10 instance cannot cross the boundary of the Windows instance.

```
dphillips@0x0002:~|⇒ tpm2_takeownership -l "hex:9da86e6eadc1d07981e03e25ab1f429feacad511" -L "hex:8cde1e89ecc16226b2b4206f7f4796e28c5765c8" -V
INFO on line: "114" in file: "tools/tpm2_takeownership.c": Successfully changed hierarchy for Lockout

dphillips@0x0002:~|⇒ tpm2_takeownership -c -L "hex:9da86e6eadc1d07981e03e25ab1f429feacad511" -V
```

Figure 5: Changing a lockout authorization value set in Windows 10

If a given TPM device is provisioned in Linux (see Figure 1), the owner authorization value set in Linux cannot be used in the context of Windows 10. We observed that in such a scenario, Windows 10 does not re-provision the TPM. In addition, the non-programmatic TPM configuration capabilities of Windows 10 ((ERNW_WP5), Section 3.1.2) do not enable users to manually input the owner authorization value in order to execute protected TPM commands and functionalities (e.g., changing the set owner authorization value). This is because a Windows 10 instance does not consider a TPM device trustworthy when the ownership of it has been taken in another operating system instance.

2.1 User Manual

This section provides step-by-step instructions for provisioning a non-initialized TPM outside of the Windows operating system with `tpm2-tools` in Linux. The TPM provisioning process involves taking ownership of the TPM and storing an SRK in the TPM device. It is relevant to emphasize that if the TPM has already been provisioned in Windows 10, it first must be cleared by the firmware so that the TPM provisioning process in Linux succeeds. The TPM provisioning approach presented in this section involves only the password-based authorization type (see Section **Error! Reference source not found.**). The `tpm2-tools` command-line argument presented as part of this approach are relevant for taking ownership of the TPM and storing an SRK in it. Taking ownership of the TPM means storing authorization values for password-based authorization.

Step 1. Use `tpm2_getrandom` to generate a strong random byte sequence that will be used as an authorization value.

Synopsis:

```
tpm2_getrandom [argument]
```

The output defaults to the `stdout`.

Example: Generate a random 20-byte sequence and convert the output data to hex without a leading "`0x`".

```
[tpm2_getrandom 20 | xxd -ps]
```

Step 2. Use `tpm2_takeownership` to store the random byte sequence generated in Step 1 in the TPM as an authorization value.

Synopsis:

`tpm2_takeownership [options]`

The random values provided to `tpm2_takeownership` can be in two forms: string and hex-string. By default, `tpm2_takeownership` assumes that provided values are in string form. The value form can be specified with the following prefixes:

`str`: indicates a value in string form.

`hex`: indicates a value in hex-string form [example: "`hex:8079cd2e066b53c0`"].

Options:

`[-o,-owner-password=ownerAuth]` stores a value as an `ownerAuth`.

`[-e,-endorse-password=endorsementAuth]` stores a value as an `endorsementAuth`.

`[-l,-lock-password=lockoutAuth]` stores a value as a `lockoutAuth`.

Example: Store the hex-string values `801`, `802`, and `803` in the TPM as an owner, endorsement, and lockout authorization values, respectively.

```
[tpm2_takeownership -o "hex:801" -e "hex:802" -l "hex:803"]
```

Step 3. Use `tpm2_createprimary` to generate an asymmetric key that is to be stored in the non-volatile memory of the TPM as the SRK.

Synopsis:

`tpm2_createprimary [options]`

This command generates a key under the owner, platform, or the endorsement hierarchy (see Section **Error! Reference source not found.**).

The output of this command includes the address (in the context of the volatile memory of the TPM) at which the generated key is stored.

Options:

`[-H,-hierarchy=object]` specifies the hierarchy under which the key is generated: `o` specifies the owner hierarchy, `p` specifies the platform hierarchy, and `e` specifies the endorsement hierarchy.

`[-P, -pwdp=auth]` specifies the authorization value for the hierarchy specified with `-H`.

`[-g,-hash=hashAlgorithm]` specifies the hash algorithm for hashing the name of the key assigned by the TPM.

`[-G,-kalg=keyAlgorithm]` specifies the algorithm type for the key to be generated

Example: Create a key under the owner hierarchy.

```
tpm2_createprimary -H o -g sha256 -G rsa -P "hex:801"
```

Step 4. Use `tpm2_evictcontrol` to store the key generated in Step 3 in the non-volatile memory of the TPM, under the owner hierarchy (see Appendix, “Persistent Objects”). This is effectively the SRK.

Synopsis:

`tpm2_evictcontrol [options]`

This command stores or removes a key in, or from, the non-volatile memory of the TPM.

Options:

`[-A,-auth=Auth]` specifies a hierarchy: `o` specifies the owner hierarchy, `p` specifies the platform hierarchy, and `e` specifies the endorsement hierarchy

`[-P, -pwdp=auth]` specifies the authorization value for the hierarchy specified with `-A`.

`[-H,-handle=]` specifies the address (in the context of the volatile memory of the TPM) of the key generated in Step 3 (see Step 3)

`[-S,-persistent=persistentHandle]` specifies address (in the context of the non-volatile memory of the TPM) at which the key generated in Step 3 should be stored. These addresses are pre-defined (see “SRK” in the Appendix, “Persistent Objects”)

Example: Store a key in the non-volatile memory of the TPM, at the address `0x81000001`.

```
tpm2_evictcontrol -A o -H 0x80000000 -S 0x81000001 -P "hex:ownerAuth"
```

Appendix

Persistent Objects

Object name	Reserved addresses
EK	0x81010001
EK Certificate	0x01C00002
SRK	0x81000001
IDevID Key	0x81020000
IDevID Certificate	0x01C90000

Table 1: Pre-defined addresses for storing objects

References

ERNW_WP5. (kein Datum). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 5.

ERNW_WP8. (kein Datum). *SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 8*.

git_tpm2tools. (30. 08 2019). Von https://github.com/tpm2-software/tpm2-tools abgerufen

nist_sp_800-147b. (11. 11 2019). Von https://csrc.nist.gov/publications/detail/sp/800-147b/final abgerufen

tcgtpm. (30. 08 2019). Von https://trustedcomputinggroup.org/resource/a-practical-guide-to-tpm-2-0/ abgerufen

Keywords and Abbreviations