CHAPTER 16

Auditing TPM Commands

As used in the TPM, *audit* is the process of logging TPM command and response parameters that pass between the host and the TPM. The host is responsible for maintaining the log, which may be in host memory or on disk. An auditor can later use the TPM to attest to the log's integrity (that it has not been altered) and authenticity (that it was logging TPM transactions).

The underlying audit concept is similar to that of attestation using PCRs. The TPM extends command and response parameter hashes into an audit digest. The auditor can later request a signed audit digest and verify the signature and certificate chain. The auditor can then walk their local copy of the audit log to validate its integrity.

Audit always records both command and response parameters and only audits a successful command. The latter requirement vastly simplifies an implementation.[1]

This chapter first gives a rationale as to why you may want to audit, then describes the audit types, and finally goes on to the details of the audit mechanism.

Why Audit

Why would an auditor want a certified list of command and response parameters? This section provides several use cases, from auditing a single command to auditing an atomic sequence of commands to auditing a continuous stream of commands.

In the simplest case, the TPM can audit one command. In a sense, this is a generalization of a TPM quote, which signs PCRs. In fact, you could do a quote using audit: start an audit log, read a set of PCRs, end the audit log, and request a signed audit digest. Although it's simpler to use `TPM2_Quote` for a PCR attestation, it can't be used to quote NV PCRs.[2] Audit is the only way to quote an NV PCR.

While there is already a TPM point solution for getting a signature over PCRs, audit provides a slightly more complicated but more flexible facility.

[1]Historically, TPM 1.2 did not at first have this requirement. This led to many corner cases where the failure was itself due to the audit, where the audit digest was partially updated but then the command failed, and so on. Late in TPM 1.2, the requirement was changed to "only audit successful commands," and this was carried forward to TPM 2.0.

[2]See Chapter 11 for an explanation of how to create an NV extend index PCR.

USE CASE: WHAT TPM AM I CONNECTED TO?

An auditor wants to know the precise TPM properties: manufacturer, firmware revision, and so on. The auditor starts an audit log, runs several `TPM2_GetCapability` commands to read the properties of interest, and then validates the audit log to ensure that the responses are legitimate.

The TPM commands are as follows:

- `TPM_StartAuthSession`: Start a session to be used for audit
- `TPM2_GetCapability`: Set the audit attribute, read the manufacturer and firmware version, and keep a log of the results
- `TPM2_GetSessionAuditDigest`: Get a signature over a digest of the log. The auditor uses the signature to verify that the log containing the capabilities has not been tampered with.

USE CASE: WHAT IS THE STATE OF AN NV INDEX, COUNTER, OR BIT-FIELD INDEX?

These indexes might be used to revoke the use of another entity through the entity's policy. That policy would use the `TPM2_PolicyNV` command, where the NV index is either a counter or a bit field. Chapter 14 explains the policy use case. Here, the caller is concerned that the NV index might not have been updated correctly. For example, the caller wants to ensure that a counter has been incremented or a bit set in a bit field. The caller can audit a read of this index to get a signed digest of its value. In some cases, where the index is authorized using an HMAC, the response HMAC itself provides response integrity. However, if the index is password or policy authorized, or if the caller doesn't have the HMAC key, audit provides the required integrity.

The TPM commands are as follows:

- `TPM2_StartAuthSession`: Start a session to be used for audit
- `TPM2_NV_Read`: Set the `session` attribute and read the index being used in the policy. The caller keeps an audit log.
- `TPM2_GetSessionAuditDigest`: Get a signature over a digest of the log. The caller uses the signature to verify that the audit log containing the NV read data has not been tampered with.

USE CASE: NV INDEX USED AS A PCR

As described in Chapter 11, a hybrid extend index can be used to implement PCRs beyond the platform-specified value. These can't be attested to using the `TPM2_Quote` command, but a signed audit gives equivalent integrity.

The TPM commands are as follows:

- `TPM2_StartAuthSession`: Start a session to be used for auditing.
- `TPM2_NV_Read`: Set the `session` attribute and read the index being used as a PCR. The caller keeps an audit log.
- `TPM2_GetSessionAuditDigest`: Get a signature over a digest of the log. The caller uses the signature to verify that the audit log containing the NV PCR value has not been tampered with.

Audit Commands

This is a summary of the TPM commands used for audit. See the TPM 2.0 specification Part 3 for the complete command set and API details:

- `TPM2_StartAuthSession` is used to start a session that can be used for audit.
- `TPM2_GetSessionAuditDigest` returns the session audit digest and optionally a signature over the digest.
- `TPM2_GetCommandAuditDigest` returns the command audit digest and optionally a signature over the digest.
- `TPM2_SetCommandCodeAuditStatus` determines which commands are included in a command audit digest.

Audit Types

The TPM library supports two audit types: command audit and session audit.

Command Audit

Command audit has two important traits, which it shares with TPM 1.2 audit.

First, it's on a per-command basis. Most commands include an attribute that, when set, indicates that the TPM should audit all instances of the command. There is a global, TPM-wide audit digest, and an auditor can request a signature over that digest.

Second, it's optional in the PC Client TPM specification. In TPM 1.2, to keep down development and test costs, vendors routinely ignored optional commands. Hardware 1.2 TPMs didn't implement command audits. Software can't rely on command audit being implemented in all TPM 2.0 devices.[3]

USE CASE: AUDITING THE TPM USED AS A CERTIFICATE AUTHORITY

A TPM can be used as a certificate authority (CA). As a hardware security module, it protects its private signing key far better than a software solution. A CA might want a verifiable list of all certificates that it signed. By setting a command audit of the `TPM2_Sign` command, the auditor can verify the list of signatures and detect any tampering of the list.

The TPM commands are as follows:

- `TPM2_SetCommandCodeAuditStatus`: Make `TPM2_Sign` be audited.
- `TPM2_Sign`: Uses the TPM as a CA to sign certificates. The caller keeps an audit log.
- `TPM2_GetCommandAuditDigest`: Gets a signature over a digest of the log containing the certificate hashes that were certified. The caller can use the signature to verify that the audit log has not been tampered with.

Session Audit

Session audit is new for TPM 2.0. It's mandatory in the PC platform specification, so it's likely to be widely available.

As the name suggests, session audit provides for an audit digest per session. An authorization session can additionally be used as an audit session by simply setting the `audit` attribute in each command to be audited. That is, a session doesn't become an audit session at the time it's started, but rather when it's used with the `audit` attribute set. For commands that don't require authorization, or to decouple audit from authorization, the audit session can be a separate session.

For example, `TPM2_Create` requires one authorization session to authorize the parent key. This session can also be marked as an audit session. Alternatively, a second session can be included with the command, this one marked for audit. `TPM2_GetCapability` requires no authorization and is normally used with no sessions. However, a session can be used for audit.

A command with multiple sessions can mark only one as an audit session.

[3] `TPM2_GetCapability` with the parameter `TPM_CAP_COMMANDS` retrieves a list of implemented commands.

Audit Log

The beginning of the chapter said that command and response parameter hashes are logged on the host, and the auditor can validate the signed log. This section outlines the required steps:

1. The auditor retrieves a list of command and response parameters from an audit log that the host stored as it executed the commands. From these, command parameter and response parameter hashes are calculated.

    a. Fortunately, the command-parameter and response-parameter hash calculations used for audit are exactly the same as those used for authorization. The command and response parameters are serialized (marshaled), and a digest over the resulting byte stream is calculated.

    b. For a command, the hash calculation, which requires marshaled parameters, should be straightforward. A TSS would naturally expose command-parameter marshaling to assist in the command-parameter authorization operation. Responses are trickier, because a TSS naturally unmarshals responses but doesn't marshal them. One approach is for the audit log to hold the marshaled response as well as the response parameters. The auditor can use the TSS to unmarshal and then validate those response parameters. If the TSS doesn't expose the unmarshal function, or if the audit log doesn't hold the marshaled response, the auditor has no choice but to write or obtain a marshaling function. Because a TPM naturally has this function, it's possible that it can be copied from a future open source TPM implementation.

    c. Either way, at the end of this step, the auditor should have command and response parameter hashes that are cryptographically validated against the command and response parameters.

2. The auditor performs the equivalent of an extend calculation, accumulating each command plus response parameter hash from step 1 into an audit digest.

3. The digital signature is verified. The calculated audit digest from step 2 is validated against the TPM signature and a public key.

4. The auditor walks a certificate chain back to a trusted root certificate, thereby establishing trust in the verification public key.

For continuous auditing, it's likely that the public key will be cached.

USE CASE: USING THE TPM TO SECURE AN APPLICATION AUDIT LOG

In addition to auditing TPM functions, the TPM audit facility can secure an application audit log. The application creates an NV extend index to record its events. Each time it records an event, it first extends that event into the NV index. It later gets a signature over the NV index data and uses it to verify that the event log has not been tampered with.

The TPM commands are as follows:

- `TPM2_NV_DefineSpace`: Define a hybrid extend index
- `TPM2_NV_Extend`: Extends the application event while also recording the event in the application event log.

When the application wishes to validate the audit log:

- `TPM2_StartAuthSession`: Starts the audit session
- `TPM2_NV_Read`: Reads the event digest
- `TPM2_GetSessionAuditDigest`: Gets a signature over the NV read data

If available, `TPM2_NV_Certify` can be used to get a signature over the NV read data, but that command may not be present on all TPMs.

Audit Data

The session audit digest is read using the `TPM2_GetSessionAuditDigest` command. In the typical use case, a signing key is supplied and the response is signed.

The digital signature isn't merely over the audit digest. As with other attestation functions, the TPM wraps the digest in a structure that includes other information. The TPM specification Part 2 describes this wrapping, where a `TPMS_ATTEST` wraps a `TPMU_ATTEST` union, which is a `TPMS_SESSION_AUDIT_INFO` structure.

The `TPMS_ATTEST` fields were covered in Chapter 12, including `TPM_GENERATED`, the qualified name of the signing key, the "extra data," the clock, and firmware information. Their security properties are the same here.

`TPMS_SESSION_AUDIT_INFO` includes, as expected, the session audit digest. It also includes a flag indicating the "exclusive" status of the session. See the following section.

Exclusive Audit

Exclusive audit permits an auditor to validate that a sequence of commands in an audit log was contiguous—that no other commands were interleaved with the exclusive sequence. A caller can designate only one session as an exclusive session. The caller sets the audit session `auditExclusive` attribute as part of a command. Assuming there was no

exclusive session already in progress, this session becomes the exclusive session, and the attribute is echoed in the response.

Once a session becomes the exclusive session, it can be used for several commands. However, any intervening command not using this exclusive audit session causes it to no longer be the exclusive session. That is, an exclusive session in progress doesn't block another command but does record that another command intervened.

When the audit digest is returned, the structure includes a flag, `exclusiveSession`, which is true if there were no intervening commands.

USE CASE: ENSURE THAT PCRS DO NOT CHANGE DURING A COMMAND SEQUENCE

A user wants to run a sequence of commands at a specific trust state. PCR values indicate the trust state of the platform. The user therefore wants to ensure that PCR values don't change during a sequence of commands. The user runs the sequence in an exclusive session. If there was a PCR extend between two commands, it changes the current exclusive session. When the caller next tries to use the original exclusive session, the TPM returns an error, indicating an intervening command.

The TPM commands are as follows:

- `TPM2_StartAuthSession`: Starts a session to be used for the exclusive audit.
- TPM command sequence that should be run without an intervening PCR extend. Set the `audit` and `auditExclusive` session attributes.
- If there was an intervening command, the request for an exclusive audit session returns `TPM_RC_EXCLUSIVE`.

Summary

Audit in the TPM is the process of logging command and response parameters. The TPM logs these parameters with an extend operation, similar to that used for PCRs, while the host saves the actual parameters. Later, the TPM can return a signed digest of the audit log. The recipient can validate the signature and thus verify the integrity of the log.

The TPM offers two audit options. Command audit records all instances of a selected group of commands, regardless of the session. Session audit records all commands in a session, regardless of the command. An exclusive session permits the recipient to detect whether an audit session was interrupted by an intervening, non-audited command. It can also provide a guarantee that there was no intervening command.