Chapter 15

Advanced Security Mechanisms in Intel® Active Management Technology

Reason and free inquiry are the only effectual agents against error.

—Thomas Jefferson (1743–1826),
Notes on the State of Virginia

In this chapter we cover some of the security protections built into Intel® Active Management Technology (Intel AMT) that were not there in its first version, but were added in the later versions to make it more secure and resilient to attacks. These include some new hardware features as well as firmware features. The new hardware includes features such as a true random number generator, chipset key, and monotonic counters. The new firmware features include secure storage of sensitive data, measured launch of Intel AMT firmware and secure audit logging.

True Random Number Generator

Many cryptographic algorithms and mechanisms make use of random numbers, including several of the Intel AMT mechanisms described previously. The important feature of a random number generator (RNG) is its entropy. Entropy is the measurement of the inability of an external viewer to predict the next number that will be generated by the RNG, even if the viewer knows all the previously-generated random numbers by that

generator. Many implementations use a pseudo-RNG (PRNG), a deterministic algorithm that produces the next random number based on the current generator's state. These algorithms maintain a high level of entropy, as long as the initial state (also called "the seed state") of the PRNG is not known [1]. For example, some PRNG implementations seed themselves according to the value of one of the platform clocks. This value is considered to be somewhat unpredictable (due to the high resolution of the clock), and therefore makes a reasonable seed for the PRNG that is suitable for applications requiring a moderate level of security. However, given that a large number of platforms power up at the same time, a time that may be known to within a few minutes or seconds, this could help a potential attacker to narrow down the possibilities and therefore crack the PRNG seed state, thereby predicting the next numbers generated by the PRNG. Conversely, an attacker could learn from the numbers generated by one hacked platform to break other platforms in the enterprise (known as a BORE attack: "Break Once, Run Everywhere").

Intel AMT hardware (beginning with Intel AMT 3.0) contains a true random number generator (TRNG) hardware device, as shown in Figure 15.1. The TRNG is based on two resistors that produce a thermal noise. The noise is amplified and provided as input to a frequency-modulated low-frequency oscillator. Combined with a high-frequency oscillator, a nearly-random bitstream is produced. A voltage regulator controls the above hardware components to avoid any bias based on voltage. In addition, a logic block attempts to correct the bitstream of any bias that may have been inserted (for example, due to the non-perfect duty cycle of the oscillator), by using a standard anti-bias correction algorithm.

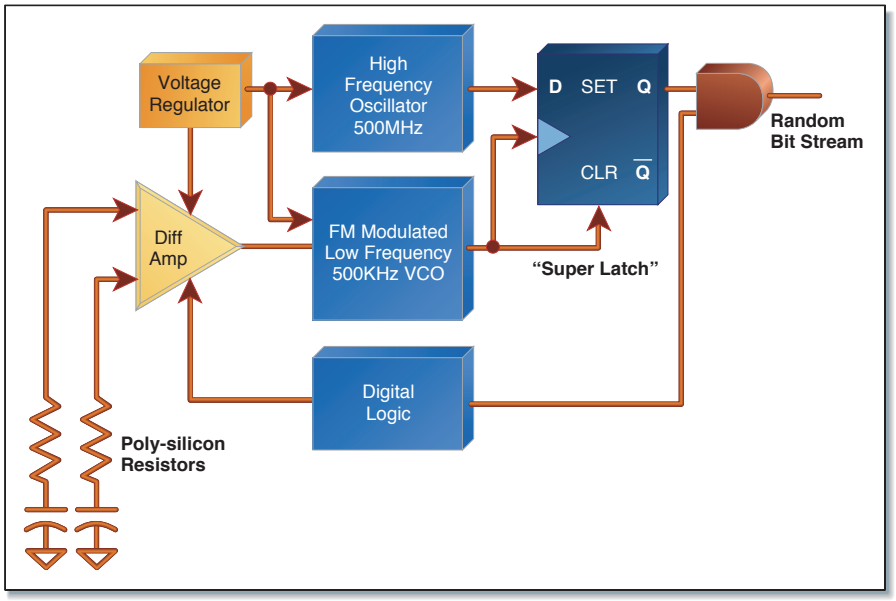

Figure 15.1 True Random Number Generator (TRNG) Hardware in Intel® AMT

One reason why it's not preferable to use this TRNG for Intel AMT usages (such as in TLS) as is, is that its takes relatively longer than a PRNG to generate random bits. In reality, Intel AMT uses a PRNG, whose state is occasionally reset, to initialize to a state generated by the TRNG. This creates a powerful high-quality RNG that is able to keep up with the high usage of random numbers in the Intel AMT subsystem.

Secure Storage of Sensitive Data: The Blob Service

As mentioned in Chapters 7 and 14, the Intel AMT nonvolatile flash memory contains the configuration data for Intel AMT, which stores some of the Intel AMT secrets. The flash controller prevents software applications and drivers running on the host operating system from accessing the flash part. However, an attacker may be able to steal a platform, pull out the flash part, and read its contents by using a flash reader. In this way, an attacker could secure a backdoor to the enterprise network by reading the secrets, or even modifying them before returning the flash part to the compromised system.

The blob service is a firmware mechanism that allows for protection of sensitive data on the flash part. The pieces of information protected by the blob service are called *blobs*. In this context, protection of a blob may take the following forms:

- Encryption, to prevent the attacker from reading the content of the blob.
- Integrity, to prevent the attacker from modifying the content of the blob.
- Anti-replay protection, to prevent the attacker from reading an encrypted/integrity-protected blob (with a value known to the attacker), and later reusing it as-is while overriding a value unknown to the attacker.

Chipset Fuse Key

The Intel AMT hardware (beginning with Intel AMT 3.0) contains a key that is unique to each system, and it is known to the Intel AMT firmware only. This key is called the chipset fuse key. The chipset fuse key is actually a set of 128 fuses in the chipset. Each fuse can be blown or un-blown, corresponding to a 0 or 1 value. The status of each of the fuses (0 or 1) is determined at manufacturing. A random subset of the fuses is blown on the chipset manufacturing line, while the rest remain un-blown. Thus, a random unique value is created for each chipset. The 128-fuse set thus creates a 128-bit key.

Encryption of secrets is achieved by using standard encryption techniques, but the interesting feature is the key that is used for the encryption. The encryption key needs to be stored in some nonvolatile form, but the flash itself is obviously not a good place to store it (otherwise the attacker would first read this key from the flash and then use it to decrypt the rest of the protected data on the flash). Rather, the Intel AMT firmware derives an encryption key from the chipset fuse key, and uses this encryption key to encrypt the sensitive items being placed on the nonvolatile flash. A similar technique is used to generate the integrity key for the integrity part of the blob service. Since Intel AMT firmware is the only entity that has knowledge of the chipset fuse key, and therefore the encryption key and the integrity protection key, even if the attacker pulls out the flash part from the system and tries to read it directly, all he sees is encrypted and/or integrity protected data, depending on the protection put in place for a given data element.

Monotonic Counters

Beginning with Intel AMT 3.0, the Intel AMT hardware contains a few registers that implement simple counters. The counters are incremented by the firmware. Those registers are unique in the sense that they are powered by the platform coin battery (also known as the RTC battery, as it powers the platform real time clock). Therefore, the counters retain their value as long as the battery is functional; this is, typically in the range of a few years.

To implement anti-replay protection, the value of the counter is incremented, then appended to the blob before applying the integrity algorithm. When the blob is read by the firmware, the value of the counter in the blob is compared to the value in the register. If they match, only then is the value considered valid. As long as the counter register is not reset (either by wraparound of the counter or by the replacement of the coin battery), the value of the counter is unique and therefore the anti-replay is achieved.

The algorithm described here requires a separate hardware register for each blob that needs to be anti-replay protected. In fact we can take this method one step further. Only one blob (let us call it "the counter blob") in the system will be protected by one hardware counter only. But the counter blob can contain counter values for other blobs. Whenever an anti-replay protected blob is modified, its private counter is incremented; this means that the counter blob is modified, which requires the incrementing of the hardware counter. Therefore, the counter blob helps us reduce several counters to a single counter, maintained by the hardware and protected by the coin battery.

When the battery is replaced, the anti-replay protected blobs are invalidated. In some cases, this will require the user to reinsert some of the secrets protected by the anti-replay blob service.

Note that we assume that every anti-replay protected blob is also integrity protected. This assumption makes perfect sense—an anti-replay blob contains a unique value that prevents it from being replayed. If the blob is not integrity-protected, the unique value can be modified, and therefore the anti-replay quality is also lost.

Equipped with the aforementioned hardware tools (chipset fuse key and monotonic counter), the Intel AMT firmware offers a blob service to all of the other Intel AMT firmware modules. The blob service provides

integrity protection, encryption, and anti-replay protection of data elements for storage on the flash. The firmware modules can decide the protections required depending on the security requirements of the data being protected.

A sample list of data blobs protected by Intel AMT by using the blob service is given in Table 15.1.

Table 15.1 Sample List of Data Blobs Protected by the Blob Service

Intel® AMT Data Structure	Integrity Protected	Encrypted	Anti-Replay Protected
Usernames and hashed passwords	Yes	No	No
Permissions, Access Control Lists	Yes	No	No
Certificates	Yes	No	No
Kerberos keys and attributes	Yes	Yes	No
Private portions of asymmetric key pairs	Yes	Yes	No
Integrated TPM secrets	Yes	Yes	Yes

Measured Launch of Intel® AMT Firmware

Someone not familiar with the concept of measured launch may ask "What's the meaning of measured launch, and why do we need it?" Let us begin by trying to answer this question. As a designer of Intel AMT, I would love to be able to say that Intel AMT is free of bugs—both design and implementation level bugs. But as you know, hardly any code is bug-free. Even some of the most critical systems such as airplane cockpit software or air-traffic control software have been known to have flaws. I remember a joke that went around a while back (probably still is making the rounds): a software engineer would never fly in an airplane if he knew that it was running the code written by him. Therefore, the entire computer industry depends on a perpetual cycle of bug-fixing and issuing updated software (also known as patches) on a regular basis that fixes the known flaws. With this mechanism we at least get rid of being vulnerable to known flaws (and hope that the fixes didn't introduce any new flaws!) But finding new vulnerabilities is not an easy task. Attackers

(and researchers; who are the good guys) spend a lot of time understanding software via reverse engineering (and several other methods), and discover new flaws. The most common way software gets compromised is by being attacked by malware that makes use of known vulnerabilities in a system that was not updated with the latest updates. It is therefore a good idea to keep any software that you use updated with the latest updates, especially the security updates.

What's even worse is the inability to be able to know which version of the software is running. Bugs in the design or code make the software vulnerable to being exploited by malware. The first thing the smartest malware tries to do is to hide itself. One of the ways it could do so is by responding to any queries regarding the version number of the software as being the latest version. This could trick the software update mechanism into believing that the latest version of the software is already running, hence there is no need to update it.

One of the best known methods to render such an attack useless is to have a trusted piece of software take a measurement (that is, a cryptographic hash) of the software that is running (or about to be run) and compare the hash with a previously generated copy of the hash (or copy obtained via some trusted means) for a given version of the software. If the hashes do not match then we would know that the software version is not what it claims to be.

Now let us see how Intel AMT uses this concept to offer a measurement of the firmware that is running inside the chipset.

Intel® AMT Firmware Measurement

Software running on the host CPU (such as the BIOS or VMM loader or the host operating system) needs to be able to measure the Intel Management Engine (Intel ME) code before it starts executing. Intel AMT firmware measurement is a feature (available in Intel AMT 5.0 onwards) that provides the capability for the Intel AMT firmware to be measured (actually the entire Intel ME firmware; though referred to as Intel AMT firmware measurement throughout this chapter) into a register in the Intel ME that is directly readable (shadowed by hardware) by host software in the MEI PCI configuration space.

The other features in the chipset that come close to providing a functionality similar to firmware measurement are the signed firmware feature, and the ICH flash protection mechanism.

The signature verification on the signed firmware is done by the Intel ME (using the verification logic in the ROM code), and the measurement done during the signature verification process is not recorded anywhere in the platform for subsequent evaluation. Therefore, the host processor has no role to play in this verification, thereby leaving a host-based VMM loader or the BIOS with no capability to assess the validity of the firmware at any later point in time after platform power on. Therefore, host software cannot implement any policy enforcement to enforce certain system behavior depending on the evaluation of the Intel AMT firmware measurement.

The other problem with using just signed firmware images as an alternative to firmware measurement is that all images (belonging to a particular chipset generation) are valid on that chipset because they are signed by the Intel code signing private key (the corresponding public key hash being embedded in the ROM). Therefore, the existing firmware signature verification mechanism makes no distinction between the various versions of the firmware images that may have been produced by Intel for a given product generation/family.

The Intel AMT firmware measurement mechanism solves these issues by providing a direct mechanism for reading the firmware measurement by host-based software (such as BIOS, VMM loader, operating system, or OS agent), without imposing the burden of the knowledge of firmware address location or offsets in the flash on the host software. The architecture ensures by design that the measurement is always completed, recorded, and locked inside an Intel ME register before the firmware execution begins. Therefore, it is not possible to overwrite this measurement value after Intel AMT firmware execution begins. The overwrite protection is guaranteed by the hardware of the Intel ME. This measurement is also available throughout the time while the Intel ME is powered on. By definition, the measurement will be different for each firmware version, thereby giving the host a mechanism to read and validate the cryptographic measurement of the firmware image. So, even if there is a vulnerability (known or unknown) that somehow exists in the Intel AMT firmware, it is not possible for malware to exploit this vulnerability in a way such that it is able to modify the previously-recorded measurement value. The Intel AMT firmware measurement value is readable by the host

software (such as BIOS, VMM loader, operating system, and so on) via the PCI configuration space of the Intel® Management Engine Interface (Intel MEI) device. (The Intel AMT processor is visible to the host OS as an Intel MEI PCI device.)

Security Audit Logs

Intel vPro™ technology creates a powerful tool for the network administrator to control the network entities. However, being in possession of a powerful tool comes with risks: the risk of erroneous use of this tool, and more importantly, the risk of malicious use of this tool. Rogue insiders are becoming a real threat to worldwide governments and enterprises, as demonstrated by a recent San Francisco network lockout [2].

A legitimate insider such as a network administrator already has very powerful credentials to access sources of business critical information in an enterprise. Unfortunately, if such administrators turn against the enterprise, they become rogue insiders, and prevention of malicious use of a privileged system is nearly impossible. However, the risk can still be mitigated by using deterrence mechanisms, and this is primarily where the Intel AMT auditing capability comes into play. This capability is available in Intel AMT 4.0 onwards.

The Intel AMT audit log shown in Figure 15.2 is an internal log that captures the administrator's operations in the system, and also captures unauthorized accesses to the system. When a security breach is discovered, the audit log can assist in tracking down the administrator (or the illegitimate user) that may have caused the breach.

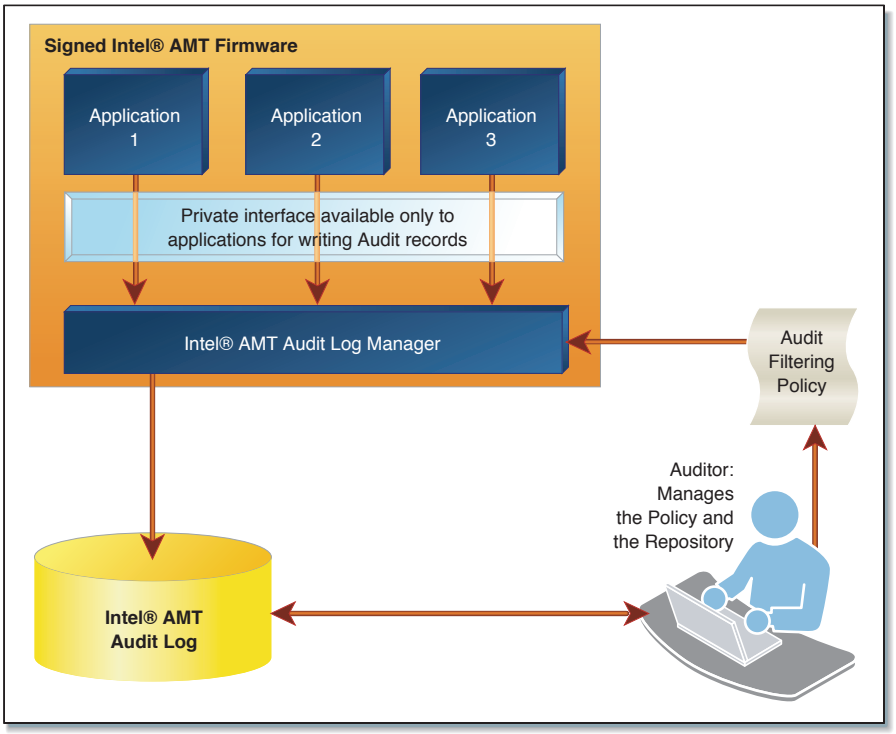

Figure 15.2 Intel® AMT Audit Log

The auditing capability cannot prevent system administrators from misusing the system, but it will prevent them from covering their tracks. The Intel AMT auditing subsystem allows an enterprise or the authorities to follow the steps of an administrator responsible for misusing the system, in a manner that is provable and undeniable. Having such a mechanism in place can deter an administrator from misusing the system in the first place.

The auditing capability also helps to provide a trail of attempts of unauthorized actions by unauthorized people or attackers, if the auditing policies have been so configured. However, this is not the primary use of the auditing subsystem, since it is assumed that the rest of the authentication and authorization mechanisms (as described in Chapter 14) would prohibit illegitimate people or attackers from performing actions within Intel AMT.

Separation of Duties

In an audited system, along with an administrator role, there is an auditor role. The auditor controls the audit log policies and contents. In many cases, an enterprise will outsource their auditing services to an impartial third-party company that provides auditing services. A separation of duties is required to create a true audited system, one that cannot be tampered with by the internal enterprise administrators. Intel AMT provides for the separation of duties by separating the roles of an administrator and auditor.

The separation of duties concept is adhered to in the credential mechanism embedded in the audit log subsystem. While the administrator might usually be omnipotent where the system is concerned, the audit log is outside of his or her realm. The administrator should not have the credentials to clear the log, modify the auditing policies, or modify the auditor's access credentials. The auditor, in turn, should only be given enough privileges to manage the audit log. Conversely, administrative operations in the system are typically out of the auditor's realm. This is the concept of *two person controls.* Thus, in order to compromise a network, and escape undetected, the administrator and auditor would have to collaborate.

The auditor role in Intel AMT is represented by an Access Control Realm called the Auditing Realm. Only the Auditor has access to this realm. Even the security administrator (the most power administrator in the Intel AMT subsystem) does not have access rights to the Auditing Realm. The Auditing Realm has commands that can export and clear the audit log and also to change audit logging policies and priorities. Any authorized administrator in the Intel AMT subsystem is allowed to read the audit log, since reading the log is not as sensitive as clearing it or changing the auditing policies.

Audit Log Records

A record in the audit log represents an administrative operation on the system. The record contains the following information:

- Identifier of the operation being logged.
- Access control credentials (username) that were used for the operation.
- IP address of the management console that initiated the operation.
- Timestamp of the operation.
- Additional information specific to the operation, if applicable.

Posting an Event to the Log

For an enterprise to claim it maintains a security log, the sequence of records in the audit log must match what transpired in the system. This means that if the administrator initiated an operation that should be logged in the audit log according to the policy, and the log entry could not be written because the log was full (an extremely rare scenario, which as we explain later, we try to prevent from occurring at all costs), then the operation fails. When the log needs to be retrieved, the auditor can be certain that no auditable operations occurred in the system other than those written in the log.

Only Intel AMT applications (which are part of the Intel AMT signed firmware) can post events to the audit log. The Audit Log Manager inside the Audit Logging component exposes a private interface to Intel AMT firmware applications to post events to the log. This prevents any outside entity from posting to the audit log and thereby trying to corrupt its integrity.

All Intel AMT applications have been designed and developed to post events to the audit log when certain operations are perpetrated, if the audit logging function is turned on.

Auditing Policy

The auditing policy defines which administrative operations should be logged in the audit log. Operations can be defined in the policy as "critical" or "non-critical." Critical events will always be logged; if a critical operation occurs and the log is full, the operation will fail. Non-critical operations will be logged only if the log is at least 20 percent empty. When the log is nearly full and a

non-critical operation occurs, the entry will not be logged and the operation will not fail. Non-critical operations are logged as space permits. The last 20 percent of the log is reserved for critical operations only.

The definition of the auditing policy is crucial to balance security with usability. When the auditor defines many frequent operations as critical, the log becomes full faster, thereby needing more frequent clearing and exporting by the auditor. On the other hand, only "critical" operations are truly audited in the foolproof sense described above.

The Audit Trail

Due to the limited capacity of the Intel AMT flash nonvolatile memory, and the usability concerns described earlier, the auditor needs to clear and export the log periodically. Before clearing the log, the auditor requests an audit trail. This is the current content of the log, signed by the firmware auditing service in a way that can later be verified by the auditor. The auditor can store the trail in long-term storage, such that if a breach occurs later, the old log can still be retrieved. The signature can attest to the fact that the logs have not been tampered with while in long-term storage. Figure 15.3 illustrates the structure of the audit log trail.

Two potential problems may arise with this approach. The first is the ability of the administrator deleting an entire signed trail from long-term storage. This issue may be addressed by adding an incremental counter to the signed trails. This allows the auditor to make sure that all signed logs are in place. In addition, the enterprise administrator should not have access to long-term storage, but a discussion of this issue is beyond the scope of this book.

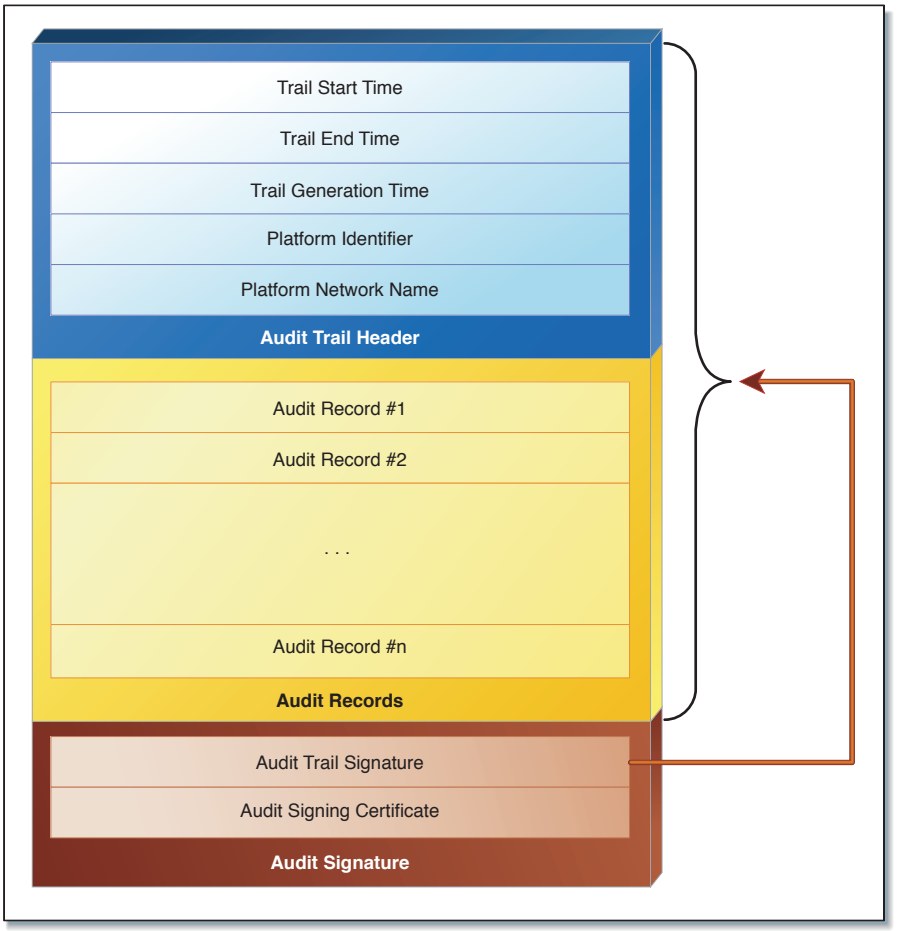

Figure 15.3 Intel® AMT Audit Log Trail Structure

A second issue that may arise from this approach is the revocation of the keying material used to sign the audit trail. It is recommended that the an additional signature be added to the auditing software by means of a temporal certificate being added to the trail before it is stored in long-term storage. When this certificate is replaced periodically, the logs in long-term storage should be re-signed. This adds another layer of authentication in case the keying material used to sign the audit trail is compromised or revoked.

This kind of an auditing system provides very robust protections against rogue-insider attacks.

Summary

In this chapter we saw how some of the more advanced security mechanisms help protect Intel AMT from attackers and malware. The true random number generator increases the security of several Intel AMT components such as higher quality session keys for TLS. The secure storage service helps to protect sensitive data on the Intel AMT nonvolatile memory from being stolen or changed by an attacker. The measured launch of Intel AMT provides assurance to an external entity that the Intel AMT firmware is indeed what it claims to be. Finally we saw how the security audit log deters insiders (such as IT technicians managing the enterprise networks) from misusing their access privileges and harming the system. It also helps to keep track of access violation attempts by adversaries who try to break in to the Intel AMT subsystem. All these security protections, along with those discussed in the previous chapter, ensure that Intel AMT is a well protected subsystem and very resilient to malware attacks.