Chapter 14

Securing Intel® Active Management Technology from Attacks

After an access cover has been secured by 16 hold-down screws, it will be discovered that the gasket has been omitted.

—De la Lastra's Corollary

The Internet is full of malware and malicious people. Any new technology is under constant threat to be attacked by adversaries for fun, fame, and profit. As a target, Intel® Active Management Technology (Intel AMT) is no different. The attackers could operate remotely and communicate with Intel AMT over the wired or wireless network interfaces. Alternatively the attacker could also operate locally by physically operating the Intel AMT computer, or by placing some malicious program in the computer's operating system that operates on behalf of the attacker. This chapter explains the details of the security protections designed into Intel AMT. These protections ensure that Intel AMT is well guarded from attacks by malicious entities (people or programs) operating remotely or locally.

However, providing robust security in any system often comes at a cost. This cost is usually in terms of reduced convenience and ease of use in using the system. This chapter also describes the tradeoffs between security and ease of use of the Intel AMT computer. It describes the various choices available to an IT architect in configuring Intel AMT such that it best suits the deployment environment, both from ease of use and security perspectives.

Threats to an Intel® AMT Computer

The previous chapters covered the unique and powerful set of security and remote management capabilities offered to the IT manager by Intel AMT. However, just like most other powerful capabilities in any system, the bad guys can and will attempt to misuse Intel AMT to attack the computer. The extent of damage caused depends on the nature of the attack. For example, it could be something relatively less severe such as causing nuisance to the end user of the computer, or something more serious such as disabling some of the security protections offered by Intel AMT.

Before going over the security protections designed into Intel AMT to defend and protect against the bad guys, it is useful to understand the threats the bad guys pose to a computer equipped with Intel AMT. Several of these attacks also apply to all computers in general.

Local Attacks

Attacks can be remote or local. A local attack to a computer means that the attacker has physical access to the computer having control of its keyboard, mouse, network connection, USB and serial ports, power button, and so on. Such an attack could take place when the user has left a computer unattended (such as when gone for lunch, or during nights if the computer is a desktop computer that the user cannot carry home at night). Of course, such an attack is one of the most powerful forms of attack, but requires the attacker to bypass many other hurdles such as the building security system, or security guards, video surveillance, and so on. A more likely scenario is that the attacker is a rogue insider. Several assumptions can be made regarding the motivations of the attacker, and the constraints under which the attacker is operating, as follows.

The attacker may simply steal the computer and walk away with it, assuming it's a laptop or a small form-factor desktop model, and not bound to the wall by a steel cable. Or he may just open the computer chassis and steal the hard disk. This assumes that the attacker is after the currently stored data on the disk, and does not worry about the discovery of the attack, since in such cases the attack will be detected pretty soon. The attacker may adopt a stealthier approach such as copying some important files from the computer's disk onto his USB flash drive. This requires the attacker to bypass the computer's login or unlock mechanisms such as BIOS password, hard-disk password, operating system or unlock password, thereby making the attack difficult.

The attacker may also be interested in installing backdoors on the computer allowing him to have subsequent access to the computer remotely. The attacker can do so by accessing the computer's disk and placing some malicious programs on the disk. This also requires the attacker to bypass the computer's login or unlock mechanisms. It also requires the malicious program to be of a nature that this undetectable by the anti-virus and anti-spyware tools installed on the computer, thereby further increasing the difficulty of the attack.

An even more invasive and hard-hitting form of local attack is possible when the attacker opens the chassis of the computer and does something really bad to the computer's hardware. Examples include installing a wireless transmitter for keystrokes, replacing the existing BIOS with a malicious version of the BIOS, replacing the firmware of Intel AMT with a malicious or modified version of the firmware, or using a hardware flash reader (available at Radio Shack for a few hundred dollars) to read sensitive data off the Intel AMT flash device. Of course, the assumption is that the attacker has access to the required tools and the time needed (with no one watching or walking by) to mount such an attack. These attacks are therefore much harder and complex, but still possible, and need to be defended against.

Several of these attack scenarios apply to Intel AMT, and the defense and protection mechanisms Intel AMT provides to protect against these attacks is the subject of this chapter. Some of the attack scenarios do not apply to Intel AMT. For example, if an attacker simply walks away with the hard disk of the computer, Intel AMT cannot prevent the data falling into the attacker's hands.

Remote Attacks

Remote attacks far outnumber local attacks because of their very nature of being mounted remotely and not requiring the attacker to be physically vulnerable at the time of conducting the attack. Remote attacks are more complex to mount due to the reduced attack surface relative to local attacks, since the remote attacks have to be mounted over the wired or wireless network connections. Following are some remote attacks that an attacker may try to mount against Intel AMT computers. Subsequently we will see how the protections designed into Intel AMT prevent these attacks.

Man in the Middle

The attacker may try to snoop the communication flowing over the network between Intel AMT and the management console. Some types of transactions between Intel AMT the management console involve some sensitive information such as security settings. Snooping can allow the attacker to have knowledge of such information. Using more complex mechanisms, the attacker may also try to intercept the communication and modify the information as it flows over the network. Such an attack can cause Intel AMT to act upon commands that were not sent by the management console but instead were sent by the attacker.

Injecting Malicious Host Software

The attacker may try to load and execute some malicious code in the host operating system. An attacker can try to inject code into the operating system space remotely in several ways. Sending malicious code via email, browser activity, and exploiting an unpatched vulnerability in the kernel are just a few examples. Once the attacker has successfully loaded his code and the code begins execution, then the code can communicate with Intel AMT in that computer using the local communication mechanisms—Intel Management Engine Interface (Intel MEI) or LMS, explained in earlier chapters—and try to attack or compromise the Intel AMT code executing in the memory of the computer, or try to attack the nonvolatile memory that stores the Intel AMT code.

Impersonation and Privilege Elevation

The attacker may try to impersonate the legitimate system administrator of Intel AMT to carry out operations within Intel AMT that would normally be allowed only by the system administrator. Password guessing, online or offline dictionary attacks, and password cracking are some ways to do this. Replaying old transactions (such as sending old network packets captured off the network) may also result in impersonation.

Typical usage of Intel AMT requires multiple system administrators to have access to various command sets in Intel AMT. For example, a security system administrator may have access to the commands that control the System Defense settings, whereas a system discovery administrator may have access to the commands for hardware asset discovery. A system discovery administrator may try to elevate his privileges to execute System Defense related commands with the malicious intention of disabling the System Defense capability of the computer.

Runtime Attacks by Exploiting Vulnerabilities

The attacker may try to exploit runtime vulnerabilities in the code of Intel AMT. Such known vulnerabilities may exist in unpatched code on computers. For example, there may be a specific vulnerability in the code of Intel AMT that can be exploited by sending a malformed network packet. Some of the common classes of attacks that exploit vulnerabilities in all types of software and firmware are buffer overflow attacks, stack attacks, and code injection. See http://en.wikipedia.org/wiki/Category:Security_exploits for details on various types of exploits.

Denial of Service

If the security protections are strong, the attacker may not be able to steal confidential information, or actively damage the computer by sending unauthorized commands. Still, an attacker may be able to do just enough so as to cause of a Denial of Service, the service in this case being Intel AMT. Examples are where an attacker can send a malformed command to Intel AMT causing it to crash. Or an attacker may alter the network settings in the routers so that Intel AMT becomes unreachable by the management consoles. Such attacks also have the capacity to cause indirect damage to the user. For example, if an Intel AMT computer cannot be contacted by the management console to perform a critical virus patch update, then the computer is at risk if the particular virus in question is at large.

Rogue Insider

Rogue insider attacks are less common than any of the aforementioned attacks, but if they occur, the consequences are catastrophic. A rogue insider is a corrupt trusted person having legitimate access rights to the system. The unique problem in this scenario is that any security measures designed into the system are ineffective by definition. There are several motivations for a trusted insider to become corrupt. Disgruntled or revengeful employees, bribed employees, or underpaid employees trying to make an extra buck by selling company secrets, and ideologically motivated employees all fall under this category. An irony of such an attack is that a trusted system administrator can easily mount the attack and also escape undetected by covering his trails. Some published case studies have revealed that rogue insiders have only been caught after a long time since they became corrupt. Often the reason for their capture has been because the corrupt employee became lazy and careless after some time, and committed some stupid mistakes that alerted others.

The following sections of this chapter and the next chapter describe the specific protections designed into Intel AMT to prevent the classes of attack described above.

Intel® AMT Process and Memory Isolation

The firmware of Intel AMT executes on the Intel Management Engine (Intel ME) in the chipset. The Intel ME, as discussed in Chapter 7, is a separate processing engine from the main CPU that executes the operating system such as Windows or Linux. Therefore, the processes running on the Intel ME are completely isolated from the main CPU and there is no direct path of process communication between the main CPU and Intel ME processes (except the Intel Management Engine Interface or Intel MEI as described in Chapter 7). The Intel MEI is very limited and provides a very specific communication channel between software in the main OS and Intel ME firmware processes such as Intel AMT.

The memory that the Intel ME firmware uses is also isolated by the chipset hardware from access by the main CPU. The chipset isolates the Intel ME memory using the UMA (Unified Memory Architecture) mechanism.

Attempts by any piece of software in the main OS to directly access the Intel ME memory is blocked by the chipset hardware.

Process and memory isolation for firmware executing in the Intel ME provides the bulk of security protection to Intel AMT from malicious software residing in the main OS. This allows the critical functionality in Intel AMT to execute unhindered, regardless of the presence of malware in the main OS space.

Intel® AMT Nonvolatile Storage Isolation

The nonvolatile storage of Intel AMT is the place where the code and data for Intel AMT is stored. In the currently shipping generations, this is a piece of NOR flash, controlled by an SPI flash controller located in the I/O Controller Hub or ICH, also known as the Southbridge. The same nonvolatile storage also stores the BIOS code and other pieces of data, such as the data used by the Gigabit Ethernet controller (GbE). Therefore, several pieces of hardware on the platform have some form of access to this nonvolatile storage. It is required that the Intel AMT code and data is not accessible by the BIOS or the GbE.

To facilitate this protection, the Intel AMT flash device is partitioned into multiple logical regions such as BIOS, Intel ME, GbE, and the Flash Descriptor, as shown in Figure 14.1. The ICH hardware defines owners for each of these regions, and defines an identifier for each owner. The owners in this case are pieces of hardware located on the platform having some sort of access to the nonvolatile storage, such as the Host CPU, Intel ME, and GbE. The ICH hardware uses the Flash Descriptor to support read/write access per owner for each region defined. At platform power on, the ICH hardware reads the Flash Descriptor data structures (located at offset 0 of the nonvolatile flash), and enforces the access control. So those region boundaries cannot be moved, nor can an unauthorized owner read/write to various regions of the flash. For example, the Intel ME cannot read/write into the BIOS region and vice versa (if the Flash Descriptor is set that way). The Flash Descriptor is itself read/write protected from all other region masters. This is the standard operating mode providing the required security for various regions of the nonvolatile flash storage device.

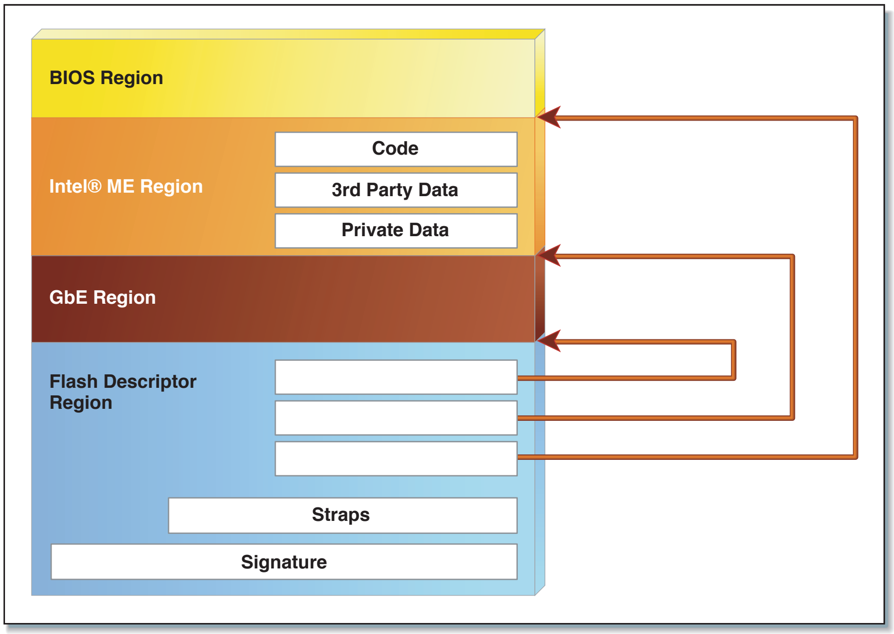


Figure 14.1 Intel® ME SPI Flash Partitions

The main purpose of the Flash Descriptor is to describe the various regions the flash device is divided into, and the different owners that can access the various regions and their read/write security permissions. Each master has direct read access only to its region. However, write access has to be explicitly granted, and is available via a hardware interface.

There is a pin (called Security Override Strap pin) in the ICH, which if set allows read/write access to the entire flash device by anyone. This is primarily provided for initial manufacturing and testing, and later on for facilitating programming or reprogramming the flash in case of service returns scenarios where, for example, the flash got corrupted for some reason. Therefore, if a malicious entity were to gain physical access to the platform, he could open the chassis of the computer and set the Override Strap, and cause a platform reset. He would then be able to gain read/write access on the complete flash part. The malicious entity could then modify the Flash Descriptor to grant itself read/write access to sensitive portions of the flash (such as the BIOS boot block), and subsequently access those sensitive portions on the flash, thereby compromising the security of the system.

However, even by mounting a physically invasive attack such as opening the computer chassis and setting the security override strap pin as mentioned above, an attacker cannot compromise the security of Intel AMT. The attacker can load a maliciously modified copy of the Intel AMT firmware on the flash device, but this copy of the firmware will not execute because the firmware signature is invalidated. The attacker cannot read sensitive information from the flash device either because all the sensitive information is secured using the blob service protection. These protection mechanisms are explained in the following sections.

Firmware Security

One of the major concerns in using software that is generally available, such as that which is available over the Internet, is the uncertainty regarding the trustworthiness of a piece of code published in this manner. There are two issues that must be addressed to make users trust the software:

- *Ensuring authenticity,* that is, assuring users that they know where the code came from
- *Ensuring integrity,* that is, verifying that the code wasn't tampered with since its publication or in transit

The use of digital signatures on the code assures recipients that the code does indeed came from the specified source.

Digital signatures are created using a public-key signature algorithm such as the RSA public-key cipher. In practice however, public-key algorithms are often too inefficient for signing long pieces of data (which is code in this case). To save time, digital signature protocols use a cryptographic digest, which is a one-way hash of the code. The hash is signed instead of the code itself. Both the hashing and digital signature algorithms are agreed upon beforehand.

Here is a summary of the process:

1. A one-way cryptographic hash of the code is produced using a standard hashing algorithm such as SHA-1 or SHA-256.
2. The hash is encrypted with the private key of the signer. The encrypted hash is the digital signature of the signer on the code.
3. The code and the digital signature (encrypted hash) are transmitted to the recipient.
4. The recipient produces a one-way hash of the code using the same cryptographic hashing algorithm.
5. Using the digital signature algorithm, the recipient decrypts the signed hash with the signer's public key.
6. If the signed hash matches the hash computed by the recipient, the signature is valid and the code is intact.

When software (code) is associated with a publisher's unique signature, distributing software over insecure media (such as the Internet) is no longer an unsafe or anonymous activity. Digital signatures ensure trust (authenticity of origin and integrity) on the code.

A good discussion on code signing is available at [9].

Intel® AMT Firmware Signing Process

Intel signs the firmware code for Intel AMT using the principles of digital code signing technology as explained above. This ensures that the only code that is executed by the Intel ME is the one that is produced and digitally signed by Intel. To provide end-to-end integrity and data origin authentication for firmware images and manifests, the Code Signing System located at Intel's secure facilities generates and stores a set of asymmetric Firmware Signing Keys (FWSK). These keys are used to generate digital signatures for manifests, as shown in Figure 14.2. The public key corresponding to the private key used to generate the manifest digital signature is placed in the manifest. Manifest digital signatures are generated using the RSA algorithm with modulus lengths of 2048 bits.

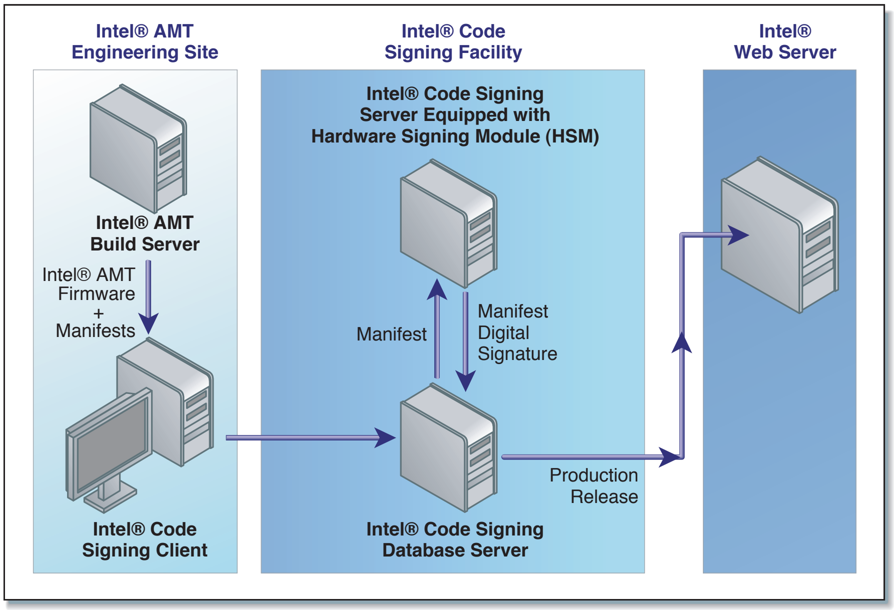

Intel® AMT Engineering Site
Intel® Code Signing Facility
Intel® Web Server
Intel® Code Signing Server Equipped with Hardware Signing Module (HSM)
Intel® AMT Build Server
Intel® AMT Firmware + Manifests
Manifest
Manifest Digital Signature
Production Release
Intel® Code Signing Client
Intel® Code Signing Database Server

Figure 14.2 Intel Firmware Signing Process

A SHA-1 hash of the public portion of the FWSK key is placed in Intel AMT system ROM. This provides a root-of-trust embedded in the chipset hardware that defeats flash substitution attacks. Therefore, an attacker cannot succeed in running a copy of Intel AMT firmware that is not signed by Intel by copying such a firmware image directly on the flash. To successfully do this, the attacker would also have to modify the public portion of the FWSK key in the ROM, which is not possible.

Asymmetric keys based on the RSA algorithm are used to eliminate the need to create system or platform unique firmware images and manifests. And because only public keys (no secrets) are stored in the Intel AMT hardware, the integrity and data origin authentication protection mechanisms for firmware cannot be compromised by a hardware attack on any single Intel AMT system.

Once a production release of new Intel AMT firmware images and manifests has been placed on its external Web site by Intel, customers such as OEMs, Enterprise IT departments, and other SMB customers can receive these updates either from Intel or from their own OEM Web sites, as shown in Figure 14.3. A Google search for Intel AMT firmware will reveal several links to OEM Web sites to obtain the Intel AMT firmware downloads. Some OEMs bundle the Intel AMT firmware image along with their BIOS images for downloading purposes.

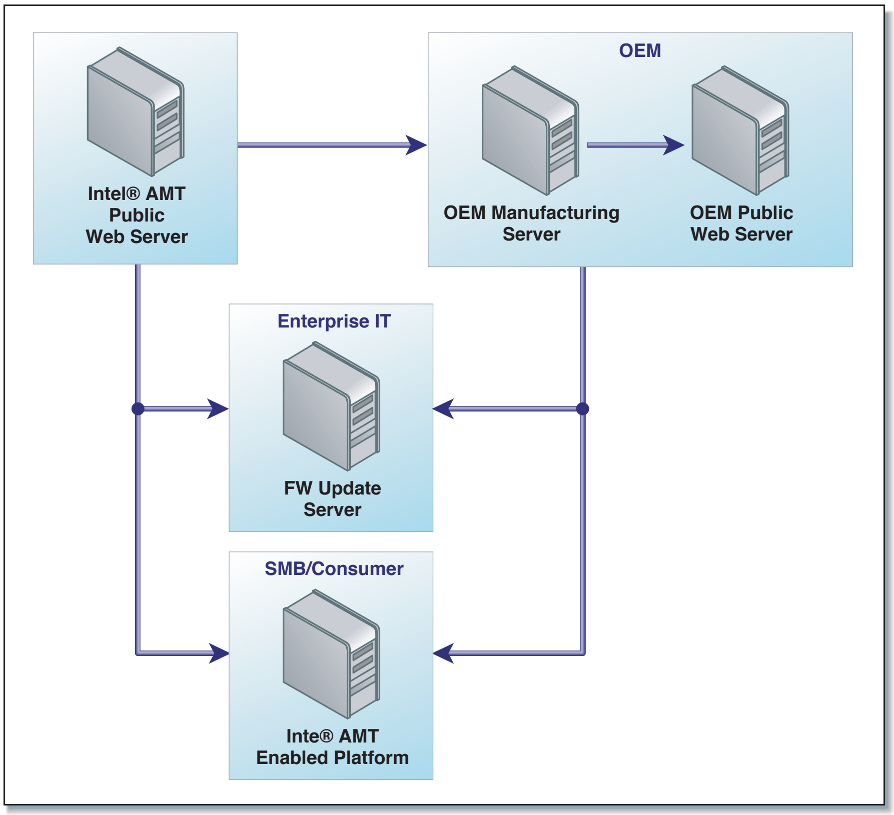


Figure 14.3 Firmware Distribution Flow

Intel® AMT Firmware Update Mechanism

Intel AMT provides a mechanism to update its firmware image. This is a very important aspect to maintain the security of the Intel AMT subsystem. The availability of a robust and secure firmware update mechanism ensures that there is a way to apply patches to the Intel AMT firmware. The patches could be for bug-fixes related to security, reliability, or the patches could contain new functionality that enhances the security of the Intel AMT subsystem. This section describes the mechanism by which Intel AMT firmware gets updated in a reliable and secure manner.

The firmware update can be done via the local interface or network interface. The local interface allows an application running in the main OS to communicate with Intel AMT and update its firmware image. Similarly a remote application can also communicate with Intel AMT and update its firmware image over the network interface. An ISV could also design a mechanism to remotely update Intel AMT's firmware remotely over the local interface by using a local agent in the main OS.

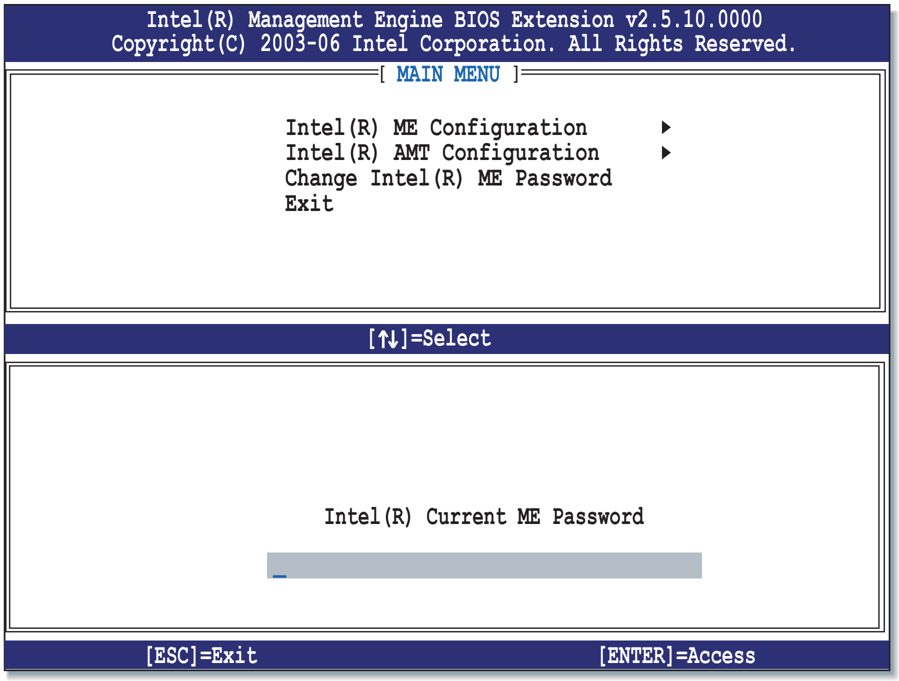
One of the problems that can occur during the firmware update process is that power to the computer can be lost (as in the case of a faulty power supply or interrupted power supply). In such a case we do not want the system to be stuck with a partial image of the firmware, thereby making it totally useless.

To mitigate this problem, the firmware image in Intel AMT nonvolatile flash memory is partitioned into two parts: a code area and a recovery area. The code area stores the main firmware image of Intel AMT, which has the firmware update application in it. The recovery area stores a small firmware application—the recovery application. These two areas of flash are never updated concurrently. Only if the update to one area is successful is the other area is updated. This ensures that there is always a working copy of a firmware update application or recovery application. The recovery application can be restarted in the event of a power failure to recover the firmware image, resulting in power-loss tolerance. All firmware needed for receiving the image and validating its integrity so as to enable standalone operation of the local recovery process is partitioned into the recovery area. The extra space in UMA is used to upload the new image, which is validated completely before being saved to flash memory. In the event of power failure or other problems, recovery code boots the next time allowing local, but not remote, recovery to take place.

The Firmware Update application in an Intel AMT subsystem operates as a service. The service listens for an incoming message originating from internal host or network client telling Intel AMT to begin an immediate firmware update across the interface specified in the message. Dual functionality of this kind ensures that Intel AMT systems can obtain the latest firmware update on their own. Alternately, when necessary, an administrator can target a specific Intel AMT system for immediate update. The Firmware Update application preserves current firmware image settings, such as OEM-provisioned data, user configured settings and date, and policy settings, enabling roll-back to a previous firmware version. Whenever firmware is written to the Intel AMT flash device, all other Intel AMT system applications are shut down and brought to a halt. Intel AMT firmware can be updated through network interfaces when the platform is connected locally in the Enterprise in S0 and S*x* states. Update can also occur in S0 through the internal host interface where both the host OS and the Intel AMT systems are operational.

Intel® AMT BIOS Security

Intel AMT has its own BIOS component that is integrated with the system BIOS. The Intel AMT BIOS component is a part of the overall Intel ME BIOS component, which is called Intel ME BIOS Extension (Intel MEBX), as explained in Chapter 7. The Intel AMT BIOS component offers several BIOS level interfaces to configure certain Intel AMT settings from the BIOS. Access to the Intel AMT BIOS screen is allowed only after providing the Intel AMT admin password. Intel AMT requires that one input the Intel AMT administrator password before proceeding to the Intel AMT BIOS screen. Figure 14.4 shows the Intel AMT BIOS password request screen. Once the correct password is provided, the MEBx proceeds to Intel AMT BIOS configuration screens.



```
Intel(R) Management Engine BIOS Extension v2.5.10.0000
Copyright(C) 2003-06 Intel Corporation. All Rights Reserved.
[ MAIN MENU ]

Intel(R) ME Configuration      ▶
Intel(R) AMT Configuration     ▶
Change Intel(R) ME Password
Exit

[↑↓]=Select

Intel(R) Current ME Password
_

[ESC]=Exit                    [ENTER]=Access
```

Figure 14.4 Intel® AMT BIOS Password Request Screenshot

After the Intel AMT admin password authentication is successful, the Intel AMT firmware allows the Intel MEBX to call into it via the Intel ME Interface (Intel MEI). For the Intel MEBX to communicate with Intel AMT firmware over the Intel MEI, no more security protection is required. This stage of availability of Intel AMT functions is called the pre-boot interface. After the Intel MEBX completes its job, it passes back control to the BIOS. After the BIOS completes its execution, it signals this event to the Intel AMT firmware. At this point, the Intel AMT firmware closes the pre-boot interface and enables the post-boot interface. The post-boot interface of Intel AMT is the one that is available over the Intel MEI to the host operating system, or over the network interface to management consoles. The post-boot interface to the host OS via the LMS route or to the management console requires authentication in the form of either HTTP Digest or Kerberos based HTTP Negotiate (discussed later in this chapter). Also, the Intel AMT firmware could be configured to mandate the use of the TLS protocol between the host operating system or management console and the Intel AMT firmware, for additional protection.

As is evident from this explanation, Intel AMT enforces a higher bar of security requirements for the post-boot interface as compared to the pre-boot interface. The post-boot security requirements must meet a higher bar because the operating system or the network infrastructure has a much wider and more exploited attack surface than the BIOS. Hence, Intel AMT invests a much higher degree of trust in the BIOS than the operating system and the network, or software running in the operating system or on the network (agents, consoles, applications, services, drivers, and so on).

Securing the Communication with Intel® AMT

Intel AMT uses Transport Layer Security (TLS) to secure its communication over the network. This protocol prevents man-in-the-middle class of attacks by providing communication security and privacy between two end-points over the Internet and enterprise intranets. The specific benefits of using TLS are as follows:

- It supports server side and client side authentication
- It is application independent, allowing other protocols like HTTP to be transparently layered on top of it
- It is able to negotiate encryption keys as well as authenticate the server (and optionally the client) before data is exchanged by higher-level applications
- It maintains the security and integrity of the transmission channel by using encryption, authentication and message authentication codes.

The TLS protocol establishes a secure channel of communication between the client and server, and consists of two parts, server authentication and optionally client authentication. Figure 14.5 shows the sequence of transactions in the TLS protocol between the client and server. In the first part, the server sends its certificate and its cipher preferences in response to a client's request. The client then generates a master key, which it encrypts with the server's public key and transmits the encrypted master key to the server. The server recovers the master key and authenticates itself to the client by returning a message authenticated with the master key.

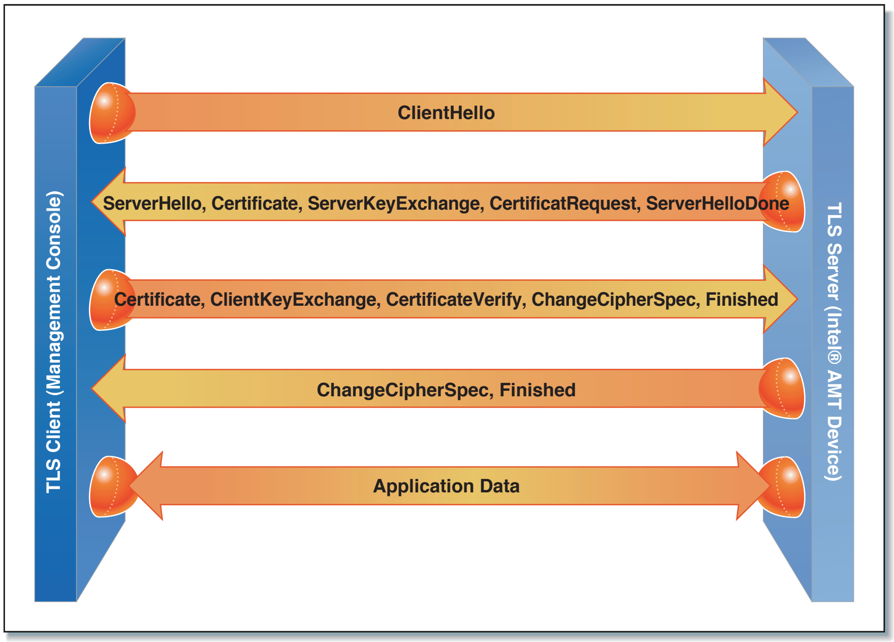

TLS Client (Management Console)
TLS Server (Intel® AMT Device)
ClientHello
ServerHello, Certificate, ServerKeyExchange, CertificatRequest, ServerHelloDone
Certificate, ClientKeyExchange, CertificateVerify, ChangeCipherSpec, Finished
ChangeCipherSpec, Finished
Application Data

Figure 14.5 TLS Protocol Handshakes

In the optional second part, the server sends a challenge to the client. The client authenticates itself to the server by returning the client's digital signature on the challenge, as well as its public-key certificate. A variety of cryptographic algorithms are supported by TLS. Subsequently, the client and server use keys derived from the master key to encrypt and authenticate the exchange of data between themselves. This was a very short description of the TLS protocol. We did not want to spend a lot of time explaining the TLS protocol in this book. The TLS protocol specification can be found in [10] and [11]. An easier description can be found in any network security book. My favorite network security book is [1].

Intel AMT works in the TLS server mode, while applications on other devices, host or management consoles, work in the TLS client mode, and initiate communications to applications on the Intel AMT system. Intel AMT supports both the mandatory first phase of authentication (server authentication) and the optional second phase of client-side authentication.

Further authentication of the IT administrator operating the Management Console (which in turn is the TLS client) is achieved using the HTTP Digest or Kerberos authentication protocols. These protocols are described later in this chapter.

To support these applications, Intel AMT supports a minimum number of simultaneous TLS sessions. As of this writing this number was 24, but Intel's engineering team fine-tunes this number with every product generation to balance functionality and performance. TLS in the Intel AMT system contains an RSA certificate or certificate chain and the RSA private key that corresponds to the leaf certificate in the chain. The public key certificate and the private keys are used for TLS server authentication during the TLS handshake.

The cipher suites and associated certificate types and key exchange algorithms supported are listed in Table 14.1.

Table 14.1 TLS Cipher Suites in Intel® AMT

Cipher Suite	**Certificate Type and Key** Exchange Algorithm
TLS_RSA_WITH_NULL_SHA	RSA, X.509v3
TLS_RSA_WITH_RC4_128_SHA	RSA, X.509v3
TLS_RSA_WITH_AES_128_CBC_SHA	RSA, X.509v3

The TLS implementation uses RSA keys with modulus lengths of up to 2048 bits. The implementation supports a single certificate hierarchy with a minimum depth of two (Root and Leaf) or one (self-signed root certificates). Certificate Revocation List (CRL) is also supported to further validate Host and Intel AMT system certificates.

The TLS_RSA_WITH_AES_128_CBC_SHA cipher suite is preferred and used whenever possible. The TLS_RSA_WITH_RC4_128_SHA is used only where the AES is not available. The TLS_RSA_WITH_NULL_SHA cipher suite will only be used where regulatory requirements do not allow the use of the confidentiality.

Authentication to Intel® AMT

The HTTP protocol provides for three authentication mechanisms: the HTTP Basic authentication, the more secure HTTP Digest authentication, and the most secure HTTP Negotiate authentication (based on Kerberos). The first two mechanisms are password-based and are detailed in RFC 2617. Kerberos is based on a symmetric key system, and is based on RFC 1510. Intel AMT supports only the latter two mechanisms, HTTP Digest and HTTP Negotiate authentication mechanisms. The reason for not supporting HTTP Basic authentication is to eliminate the risks associated with this least secure mechanism of authentication.

Password-based Authentication to Intel® AMT

HTTP Basic Authentication protocol (Intel AMT does not support this) provides for a challenge-response authentication mechanism that may be used by a server to challenge a client, and by a client to provide authentication information back to the server. In this scheme, the client sends its user ID and password to the server, and the server will authorize the client only if it can validate the user ID and password. Otherwise, the server responds with an error code. The user ID and password are sent across the wire, without any encryption, which makes the basic authentication scheme inherently insecure.

The HTTP Digest Authentication scheme is more secure than the HTTP Basic Authentication Scheme, because the password is never sent from the client to the server in the clear. In the Digest Scheme, the server challenges the client with a random value (called a nonce). A valid response contains a cryptographic checksum of the username, the password, the given nonce value, and some other data. In this way, the password is sent over the wire as a hash to prevent interception and reuse. Upon receiving the response, the server computes the checksum (a cryptographic hash) using the same inputs, and compares the computed checksum with the one received from the client. If they match, then the client is authenticated.

Intel AMT uses the HTTP Digest Authentication Scheme for authentication of the client (such as the Management Console), before allowing access to the system. A challenge is sent to the client, and a response containing the digest of the password and other information, must be returned.

The Intel AMT system stores the MD5 hash of the username, password, and the HTTP realm. The HTTP realm incorporates the Intel AMT machine ID, which is unique for every system. This makes the hash value stored on every Intel AMT system unique.

Should an attacker break into the Intel AMT system's flash memory and seizes this hash value, it is of no use in attacking other Intel AMT systems, even if the passwords of those Intel AMT systems happen to be the same. The cryptographic hashing also ensures that the passwords are hard to be reverse engineered by gaining access to the hash value. The steps involved in the HTTP Digest authentication technique are detailed in Figure 14.6.

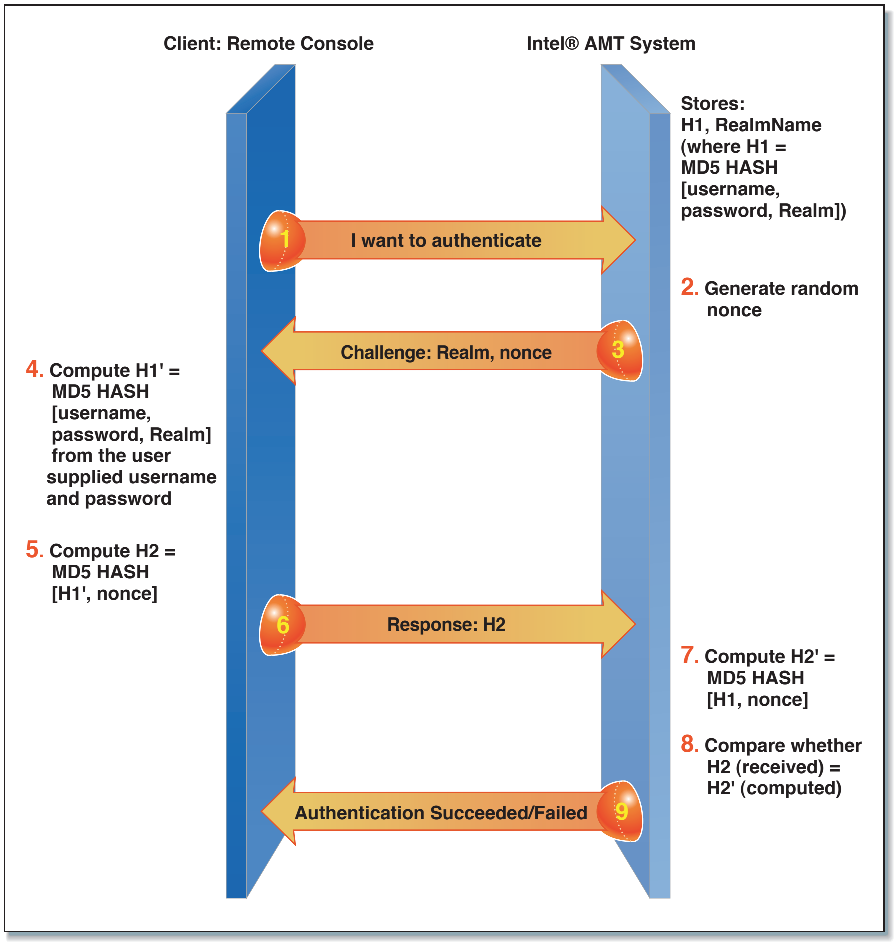

Client: Remote Console
Intel® AMT System
Stores: H1, RealmName (where H1 = MD5 HASH [username, password, Realm])
1 I want to authenticate
2. Generate random nonce
3 Challenge: Realm, nonce
4. Compute H1' = MD5 HASH [username, password, Realm] from the user supplied username and password
5. Compute H2 = MD5 HASH [H1', nonce]
6 Response: H2
7. Compute H2' = MD5 HASH [H1, nonce]
8. Compare whether H2 (received) = H2' (computed)
9 Authentication Succeeded/Failed

Figure 14.6 HTTP Digest Authentication

Intel AMT requires that HTTP Digest passwords used for authentication to the Intel AMT system meet the following minimum criteria:

- Must be at least 8 characters long, and up to 32 characters long.
- Must have at least one digit character (0, 1,…9)
- Must have at least one special non-alphanumeric character ("!", "$", "%"). Some special characters, such as double quotes, commas, semi-colons, and colons are not allowed.
- Must contain both lowercase and uppercase Latin characters

These restrictions enforced by Intel AMT help to reduce susceptibility of passwords to offline dictionary attacks by attackers.

Kerberos Authentication to Intel® AMT

Kerberos is an authentication service developed by the Project Athena team at MIT. The first general use version was version 4. Version 5, which addressed certain shortfalls in version 4, was released in 1994. Kerberos uses secret-key technology for encryption and authentication. Unlike a public-key authentication system, Kerberos does not produce digital signatures. Instead Kerberos was designed to authenticate requests for network resources.

In a Kerberos system, there is a designated site on each network, called the Kerberos Key Distribution Center (KDC), which performs centralized key management and administrative functions. The KDC maintains a database containing the secret keys of all users and machines/servers, authenticates the identities of users, and distributes session keys to users and servers who wish to authenticate one another. Kerberos requires trust in a third party (the KDC). If the KDC is compromised, the integrity of the whole system is lost. However, Kerberos is generally considered adequate within an administrative domain or enterprise. Just as a contrast, public-key cryptography was designed precisely to avoid the necessity to trust third parties with secrets.

Kerberos also addresses a key shortcoming of the password-based HTTP Digest approach. HTTP Digest requires that each Intel AMT device be configured with at least one username and password pair. In a large enterprise with thousands of systems, it is going to require a good amount of management bandwidth to configure and manage unique user-name/password pairs for all the systems. This may lead to the usage of weak passwords, or common passwords on the systems. Otherwise the ISVs will be required to develop components that effectively manage passwords, without compromising the security of the enterprise.

The integration of the Kerberos protocol over HTTP for authentication to Web services, with the Windows Domain Controller acting as the KDC, is wrapped under the HTTP Negotiate protocol umbrella. HTTP Negotiate protocol allows the use of the older Windows NTLM protocol for authentication. HTTP Negotiate is essentially a wrapper protocol over Kerberos or NTLM. Intel AMT supports authentication using the Kerberos-based HTTP Negotiate protocol. Integrating the authentication framework of Intel AMT systems on the Kerberos-based HTTP Negotiate protocol provides for a standard and single-sign-on style authentication to Intel AMT. This eliminates the need for ISV applications (including the Intel AMT configuration service) to manage unique and strong username/password pairs for all Intel AMT systems. Authentication to Intel AMT thereby becomes as strong and as secure as authentication to the Windows domain; and administrators wanting to manage Intel AMT systems need only log in to the Windows domain to gain access to Intel AMT devices. Windows infrastructure is the most widely deployed in the industry today, and integration with Microsoft Windows authentication will cater to a vast majority of the IT user base.

Note Researchers at MIT invented the Kerberos authentication protocol. The Kerberos page on the MIT Web site [2] has plenty of useful information on Kerberos. A good starting point for Kerberos documentation is [3]. My favorite is [4]. Microsoft has also developed very good documentation on Kerberos. The HTTP Negotiate Protocol is especially very nicely explained in the 3-part article [5]. Starting at [6] you can follow a gold mine of information on Microsoft Kerberos. My favorite is [7].

Following are some of the benefits derived from leveraging the Kerberos authentication that is integrated with Active Directory:

- If an IT administrator is logged into the Microsoft Windows domain using his username (that is, domain\username such as, for example usa/john) and password, then he is able to automatically authenticate (behind the scenes; without supplying any other password) to Intel AMT computers.

- An IT administrator is allowed or denied privileges to manage an Intel AMT computer based on his membership to a group in Active Directory. This will ensure that when an IT administrator is no longer supposed to manage one or more Intel AMT systems, his privileges to do so are revoked simply be removing his membership from the group in Active Directory.
- Intel AMT devices are able to ascertain the identity of the administrator attempting to gain access to the Intel AMT system, and be able to apply access control for that user. The notion is that not everyone who is successfully authenticated by Intel AMT is allowed access to all resources within Intel AMT. The authorization that a given user has is governed by an Access Control List (ACL) located within an Intel AMT device.

Let us see how the magic happens. Every Intel AMT subsystem has a unique identity (like every user and server has a unique identity) within the Kerberos domain, and is configured with a few unique Kerberos credentials (as described below) during the initial configuration process. The Intel AMT subsystem is represented as a Kerberized service and supports those portions of the Kerberos protocol that are specified for a Kerberized service. This is because Intel AMT is a service that administrators authenticate to, when they want to perform some Intel AMT actions. Specifically, the Kerberos credentials configured into Intel AMT are as follows. More details of the initial configuration process of Intel AMT are described in Chapter 17.

- Kerberos Service Principal Name (SPN). The SPN uniquely identifies the Intel AMT object's within the Kerberos domain. In Windows Active Directory, the Kerberos domain is the same as the Windows domain. The SPN takes the form HTTP/<fqdn>:<port number>, where the FQDN (Fully Qualified Domain Name) is the FQDN of Intel AMT, and the port number is the port at which Intel AMT Kerberos service runs. The port number is used to differentiate the Intel AMT Kerberos service from any other Web service running on a different port on the host operating system.

- Realm Name. This is the name of the Kerberos domain (also referred to as realm), which is the same as the Windows domain when Intel AMT is deployed with Microsoft Active Directory.
- Encryption Algorithm used by Kerberos. Usually this is RC4-HMAC.
- Kerberos Master Key for an Intel AMT subsystem. This is a unique and secret key that belongs to each Intel AMT subsystem, and is shared between the Intel AMT subsystem and the KDC.
- Clock Tolerance. This is the tolerance (usually set to a few minutes) within which the time must be synchronized between the KDC, Intel AMT and the Management Console.

With these credentials, the Intel AMT subsystem is ready to use Kerberos to authenticate administrators connecting to Intel AMT to perform Intel AMT operations. Figure 14.7 shows the Kerberos protocol at a high level. The Management Console acts as a Kerberos Client, and Intel AMT as a Kerberized Service. The Windows domain controller is the KDC in an Active Directory–based deployment. The steps shown in green in Figure 14.7 are performed by the IT administrator or the console on his behalf. Steps in red along the arrows depict the Kerberos messages that flow on the network as part of the Kerberos protocol. The steps in red towards the right of the KDC and the Intel AMT computer depict the operations performed by the KDC or Intel AMT respectively during the Kerberos authentication process. The conceptual steps that take place behind the scenes are as follows (in reality a lot more goes on, but we will leave it as an exercise for the reader to investigate the gory details).

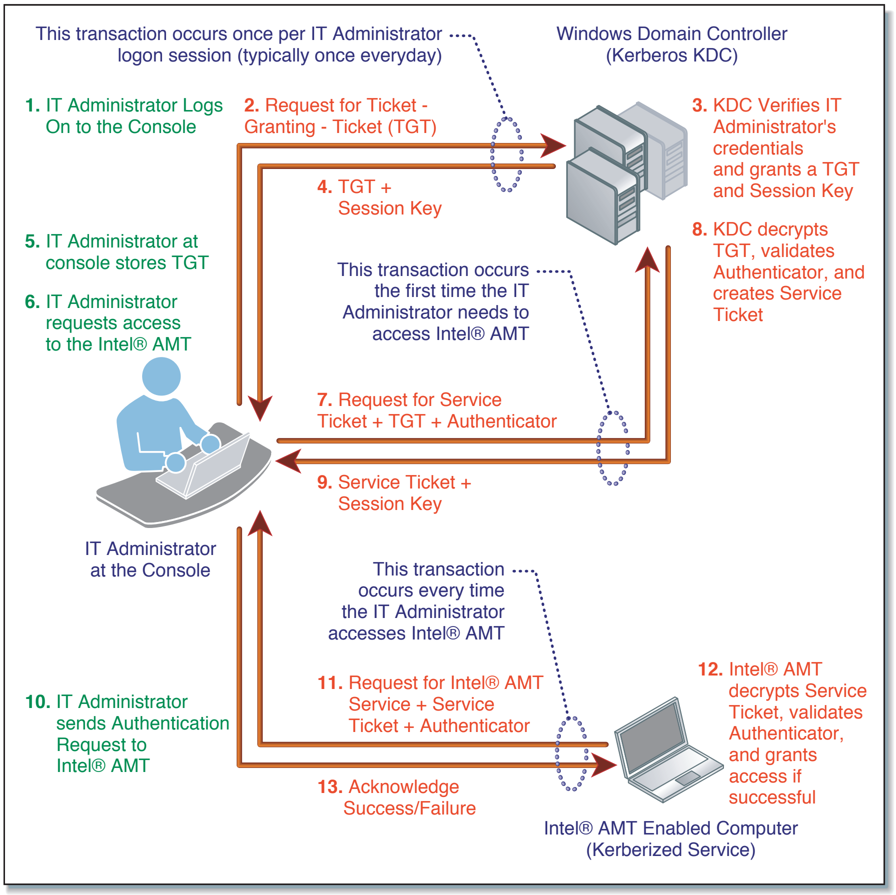

This transaction occurs once per IT Administrator logon session (typically once everyday)
Windows Domain Controller (Kerberos KDC)
1. IT Administrator Logs On to the Console
2. Request for Ticket - Granting - Ticket (TGT)
3. KDC Verifies IT Administrator's credentials and grants a TGT and Session Key
4. TGT + Session Key
5. IT Administrator at console stores TGT
6. IT Administrator requests access to the Intel® AMT
This transaction occurs the first time the IT Administrator needs to access Intel® AMT
8. KDC decrypts TGT, validates Authenticator, and creates Service Ticket
7. Request for Service Ticket + TGT + Authenticator
9. Service Ticket + Session Key
IT Administrator at the Console
This transaction occurs every time the IT Administrator accesses Intel® AMT
10. IT Administrator sends Authentication Request to Intel® AMT
11. Request for Intel® AMT Service + Service Ticket + Authenticator
12. Intel® AMT decrypts Service Ticket, validates Authenticator, and grants access if successful
13. Acknowledge Success/Failure
Intel® AMT Enabled Computer (Kerberized Service)

Figure 14.7 Kerberos Authentication Protocol

1. The IT administrator operating the Management Console authenticates to the Windows domain in the normal way. This gives a Ticket Granting Ticket (TGT) to the Management Console.
2. When the IT administrator wants to perform Intel AMT operations on a given Intel AMT computer, the Kerberos agent on the Management Console obtains a Kerberos ticket from the KDC for authenticating to Intel AMT.

3. The Management Console supplies this ticket to Intel AMT. It also supplies a token (called authenticator).
4. Intel AMT validates this ticket (because it is encrypted by the KDC using Intel AMT's master key). Intel AMT validates the authenticator as well (using a session key which is embedded inside the ticket). Intel AMT also validates that the time stamp inside the authenticator is recent (usually within a few minutes in the past). This prevents replay attacks.
5. At this time, Intel AMT has successfully authenticated the IT administrator. The IT administrator can now perform Intel AMT operations that this administrator is permitted to perform.
6. The ticket sent by the IT administrator contains a list of Active Directory groups that the IT administrator is a member of. Intel AMT uses this list to determine the permissions of the IT administrator when Active Directory–based Kerberos is used for authentication.

The next section describes how permissions are enforced on an authenticated IT administrator.

Access Control in Intel® AMT

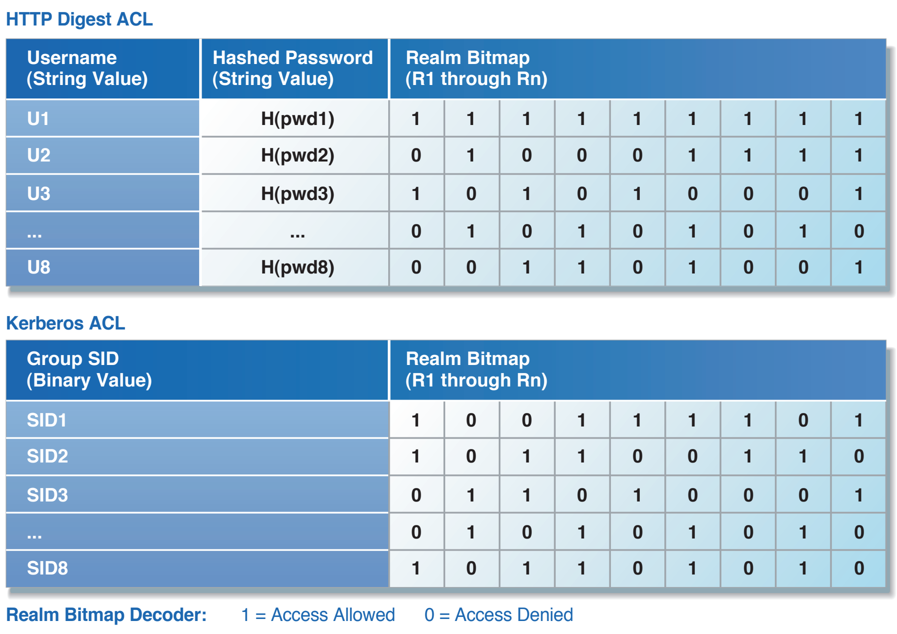
Intel AMT allows access to its operations to multiple administrators. In other words, more than one IT administrator is allowed to perform operations in Intel AMT. But not all IT administrators have the exact same privileges or permissions. Intel AMT makes it possible to group IT administrators such that one group of administrators has access to one particular group of operations; another group of administrators has access to another group of operations, and so on.

Intel AMT has predefined groups of operations. Each such group of operations is called an Intel AMT Access Control Realm. Each realm has a predefined set of operations assigned to it. Some examples of realms and the operations in those realms are given in Table 14.2. Although this list is not complete, a complete list with detailed documentation can be found in the Intel AMT Software Developer's Kit.

Table 14.2 Intel® AMT Access Control Realms

Intel® AMT Access Control Realm	**Sample Operations in the Realm**
Security Administration	Configure additional administrators, setting their credentials and permissions, configuring TLS, Kerberos, etc.
Network Administration	Configure TCPIP parameters, host name, domain name, 802.1x parameters, etc.
Hardware Asset Information	Obtain hardware information for the computer
Remote Control	Remotely control the computer by turning it on, off, restart the computer, etc.
ISV Storage	Read and write data into the ISV storage area of Intel AMT
Redirection	Configure the Intel AMT subsystem for an IDE Redirection session
Agent Presence	Register or deregister an agent presence watchdog, send a heartbeat to Intel AMT, etc.
Network Time	Get and set the Intel AMT Protected Real Time Clock
General Information	Various kinds of read operations that most authenticated IT administrators can do such as reading the version number of Intel AMT, reading the time, reading the audit log policies, reading the audit log, read the network settings, read the capabilities available in Intel AMT, read the computer's UUID, etc.
Firmware Update	Perform a firmware update for the Intel AMT firmware
Wireless Configuration	Configure wireless profiles into Intel AMT for access via wireless interface to Intel AMT
Remote Access	Configure the Remote Access parameters such as Management Presence server address, Remote Access policies, etc.
Secure Audit Log	Configure the Audit Log settings such as auditors, audit policies, export audit logs for archival, etc.
Local User notification	Allows an agent in the host operating system to be notified of certain events inside Intel AMT

As of this writing, a maximum of 16 IT administrators is configurable in any given Intel AMT subsystem, 8 based on HTTP Digest authentication, and 8 for Kerberos-based authentication. Each administrator can be assigned to one or more Access Control Realms. There is one special IT administrator who has access to all Access Control Realms. This administrator is the Security Administrator. This administrator can add or remove more administrators to the various Access Control Realms. The complete list of permissions is stored in a structure called an Access Control List (ACL). This structure conceptually looks like the one in Figure 14.8.



HTTP Digest ACL

Username (String Value)	Hashed Password (String Value)	Realm Bitmap (R1 through Rn)								
U1	H(pwd1)	1	1	1	1	1	1	1	1	1
U2	H(pwd2)	0	1	0	0	0	1	1	1	1
U3	H(pwd3)	1	0	1	0	1	0	0	0	1
...	...	0	1	0	1	0	1	0	1	0
U8	H(pwd8)	0	0	1	1	0	1	0	0	1

Kerberos ACL

Group SID (Binary Value)	Realm Bitmap (R1 through Rn)								
SID1	1	0	0	1	1	1	1	0	1
SID2	1	0	1	1	0	0	1	1	0
SID3	0	1	1	0	1	0	0	0	1
...	0	1	0	1	0	1	0	1	0
SID8	1	0	1	1	0	1	0	1	0

Realm Bitmap Decoder: 1 = Access Allowed 0 = Access Denied

Figure 14.8 Intel® AMT ACL Structure

For IT administrators using HTTP Digest, each entry in the HTTP Digest ACL structure (depicted by each row in Figure 14.8) stores the user-name, hashed password, and the Intel AMT Access Control Realms that this IT administrator has access to. For IT administrators using Kerberos, each entry in the Kerberos ACL structure (depicted by each row in Figure 14.8) stores the Active Directory Group ID and the Intel AMT Access Control Realms that members of this group have access to. Groups (and individual users) in Active Directory are represented by their SID (Security Identifier). The Intel AMT ACL therefore just lists the SID, instead of the readable string name of the group. More details on SID in Active Directory and its use for group authentication are available in [8].

For a deeper insight into access control mechanisms, refer [12]. Role based access control mechanisms (check out [13]) are also relevant to this topic.

Trusted Time in Intel® AMT

Intel AMT uses a Protected Real Time Clock (PRTC) to provide a time value. Some example usages for the need of a trusted time base in Intel AMT are as follows

- The logging application to log events inside Intel AMT with a time-stamp
- The Certificate validation process
- Kerberos ticket validation
- Execution of time-driven policies, such as checking for updates on a weekly basis

The PRTC is separate from the regular system clock that prevents users or malicious programs running with user permissions from unauthorized modification of the system time. Such improper system time changes could cause inaccurate event log timestamps, missed certification validations, or loss of Kerberos synchronization.

The PRTC is connected to the RTC (Real Time Clock) power well, so the context of the PRTC is maintained throughout all system power states. Initial programming of time on the PRTC is needed only after the installation of a new RTC battery.

Furthermore, it is necessary to prevent attackers from using a rogue time source on the network. Middlemen must be prevented from making any modifications to configuration packets that are in transit along the wire and contain time values.

Intel AMT acquires trusted time from a remote ISV Management Console. Time synchronization commands are initiated by the ISV Management Console, either periodically or as part of the discovery mechanism. For synchronizing the time, the ISV Management Console initiates a mutually authenticated TLS session with the device. Once the TLS session is established, the ISV Management Console uses a WS-Management call to set the time. The ISV Management Console keeps accurate time using the Network Time Protocol (NTP) by communicating with a NTP server, or by communicating with a Windows Domain Controller (if the Management Console is part of a Windows Domain).

Semantics that are similar to Simple Network Time Protocol/Network Time Protocol (SNTP/NTP – check out [14] and [15]) are used for calculating network latencies, resulting in high-accuracy time updates for more sensitive usage scenarios such as Kerberos. The protocol transactions are represented in Figure 14.9.

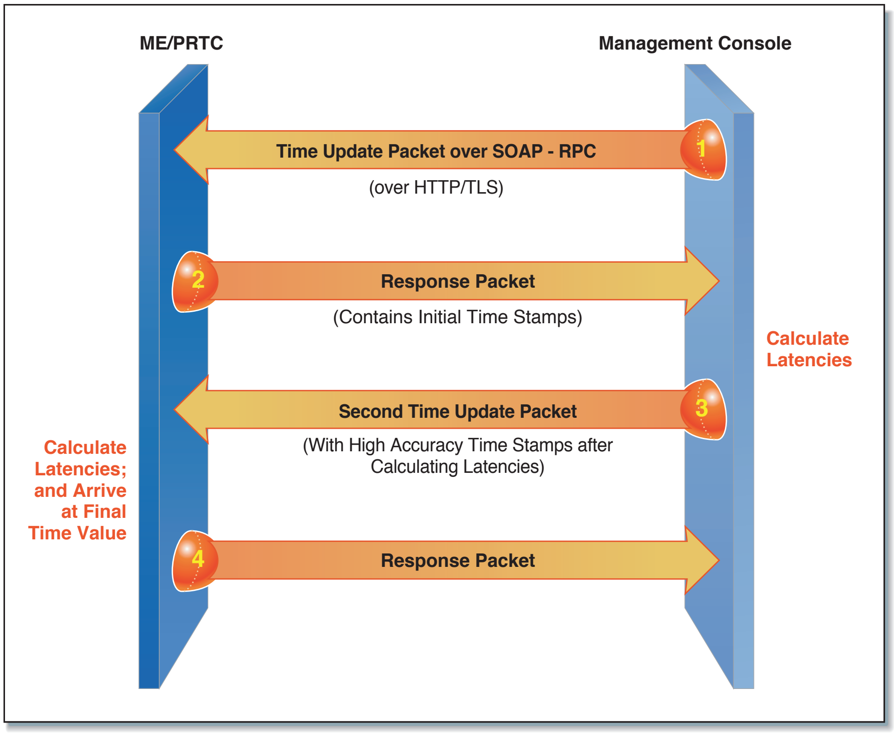

ME/PRTC
Management Console
1
Time Update Packet over SOAP - RPC
(over HTTP/TLS)
2
Response Packet
(Contains Initial Time Stamps)
Calculate Latencies
3
Second Time Update Packet
(With High Accuracy Time Stamps after Calculating Latencies)
Calculate Latencies; and Arrive at Final Time Value
4
Response Packet

Figure 14.9 Time Sync Model for ME PRTC

If for some reason the time in the PRTC goes out of sync, a network sync of time using the aforementioned protocol is required to reset the time on the PRTC. In the event that network connectivity is not available or the internal firmware for time sync process is not functioning, a BIOS command protected with Intel AMT admin login is available to set the PRTC time locally. The BIOS command is primarily a recovery mechanism. It is a good idea for the ISV Management Console to periodically sync up the time of the PRTC.

Summary

In this chapter we discussed the various protections that Intel AMT offers to ensure that the bad guys are kept out, and the good guys are let in. Table below briefly captures the protection mechanisms. In the next chapter we will go into some of the more advanced security protection mechanisms built into Intel AMT.

Table 14.3 Security Protections in Intel® AMT

Protection Mechanism	Brief Description
Separate processing engine (Intel® ME)	Protects processes in the main Operating System from directly communicating with Intel® AMT
Separate memory for Intel ME	Protects processes in the main operating system from snooping into Intel ME memory
Flash region separation	Protects Intel AMT firmware and data from being snooped or overwritten by malicious entities
Firmware Signing	Prevents execution of arbitrary firmware on the Intel ME. Digital signature check ensures that only Intel-signed firmware executes on the Intel ME
Intel AMT BIOS password	Protects malicious entities from accessing the Intel AMT BIOS screen and changing the Intel AMT configuration settings
Intel AMT Authentication	Ensures that only authenticated entities can communicate with Intel AMT
HTTP Digest Authentication	Password based authentication to Intel AMT. Prevents password to be snooped on the wire. Also prevents BORE attacks, even if the same password is used on multiple Intel AMT systems
HTTP Negotiate (Kerberos) Authentication	Kerberos-based authentication to Intel AMT, integrated with Microsoft Active Directory Group permissions. Provides Single-Sign-On to Intel AMT, thereby making it scalable in the enterprise
TLS for on-the-wire security	Protects the communication between Intel AMT and the ISV management console. Someone snooping on the wire cannot read or modify the Intel AMT commands or responses. Also authenticates the machines on the two sides to each other.
Access Control	Separates the rights of one Intel AMT admin from another.
Secure Time	Protects the time value to be changed by a malicious entity in the main OS. Provides trusted time to other Intel AMT components such as certificate verification component, Kerberos component, etc.