

Microsoft Vista: Trusted Platform Module Services

Solutions in this chapter:

- Understanding the TPM
 - Configuring and Managing the TPM on a Stand-Alone System
 - Configuring and Managing the TPM in an Enterprise Environment
 - TPM Applications
 - Understanding the Security Implications of the TPM
-
- ☑ Summary
 - ☑ Solutions Fast Track
 - ☑ Frequently Asked Questions

Introduction

The Trusted Computing Group (TCG) was formed in 2003 with the goal of developing and promoting open standards for trusted computing. The group was founded by Advanced Micro Devices, Hewlett-Packard, IBM, Infineon, Intel, Lenovo, Microsoft, and Sun Microsystems, and currently has 135 members. The main product to come out of the TCG so far is the specification for the Trusted Platform Module (TPM), and the corresponding specification for the Trusted Computing Group Software Specification (TSS). The TPM is the key hardware component of a trusted computing platform and the TSS is the specification for an application program interface (API) that developers can use to create software that will interact with the TPM.

Windows Vista supports only version 1.2-compliant TPM devices natively, although third-party software, such as Wave's Embassy Trust Suite, is available that you can use to support some functionality of version 1.1b TPM devices. Version 1.2 of the TPM specification from the TCG was published in October 2003, and has since been revised. In this chapter, we will cover only the native Windows Vista support for version 1.2 TPM devices. In 2006, PC OEMs started to make a big push to incorporate TPM version 1.2 devices into their products. Laptops seem to be on the leading edge of this new technological wave, with Dell, HP, and Lenovo cranking up production of laptops.

Understanding the TPM

The TPM is a microchip for motherboards that is designed to create, store, and protect cryptographic keys, and the fact that the TPM is integrated into motherboards allows it to be used as a tool for strong authentication of devices as well. The TCG intends for the TPM to be the foundation upon which secure computing platforms are built. Because the TPM is a dedicated piece of hardware that creates a master encryption key which can be used to encrypt other keys that are created by user mode applications, keys are separated from software vulnerabilities that would directly jeopardize the security of keys protected only by software. A key protected by software is susceptible to disclosure the moment a buffer overrun or some other flaw in the software code is discovered. Protecting encryption keys with a hardware device such as the TPM can be seen as a direct response to many of the most prevalent and serious security issues facing information security professionals today, including rising numbers of software vulnerabilities, theft of equipment containing sensitive data including personally identifiable information (PII), and identity theft.

NOTE

Personally identifiable information (PII) is any information that may be used to uniquely identify a person, such as a Social Security number (SSN), credit card numbers, a person's health information (which the Health Insurance Portability and Accountability Act of 1996 [HIPAA] was enacted to protect), and the combination of a full name and address. It is critical to protect this information because it is useful to identity thieves.

However, cryptography is about more than just encrypting large amounts of data in order to keep it confidential. Cryptography is simply the process of taking a block of plain text and processing it to create cipher text. There are few limits as to what that plain text can be and what that cipher text can be used for. A lot of different applications for cryptography have been developed, including the following:

- A cryptographic algorithm can be applied to a file (or set of files) in order to create a hash or message digest which serves as a fingerprint of the file(s). This function is widely used for gauging the integrity of data, and is critical to the field of forensics.
- Digital signatures utilize public key cryptography and message digests in order to allow recipients to verify the integrity and authenticity of messages they receive.
- Sensitive user data can be processed using cryptographic algorithms in order to protect the confidentiality of the data.

The TPM and Windows Vista TPM services use all of these functions to implement the functions they provide, and in doing so they also support users' ability to securely utilize these cryptographic functions as well. Before getting into the technical details of how this works, we'll look at what a trusted platform is, who developed such an idea, and where the TPM fits in.

Are You Owned?

Has Anyone Seen My Laptop?

In 2006, we witnessed a parade of news stories in which employees lost laptops that contained PII or other sensitive data:

- In April, Boeing announced that a laptop with PII of 3,600 of its employees had been stolen.
- In May, a U.S. Department of Veterans Affairs employee's laptop and external hard drive were stolen from his home. The laptop contained PII, including SSNs, on more than 26.5 million soldiers.
- In June, Hotels.com announced that an auditor from Ernst & Young Global Ltd. had his laptop stolen, exposing the names, addresses, and credit card information of nearly one-quarter of a million Hotels.com customers to thieves.
- In September, the U.S. Department of Commerce reported that 1,100 of its laptops were missing, and that 249 of the laptops contained PII.
- In November, Kaiser Permanente notified 38,000 members that their personal health information had been compromised when an employee's laptop was stolen out of his car.
- In December, we heard from Boeing again. This time it announced that a laptop theft had compromised the personal information of 382,000 employees.

These are just a handful of the highest-profile data loss stories from 2006, and the list of headlines for 2007 should be long as well. Not only are companies and government agencies relying more than ever before on laptops to serve their workforce, but more smartphones and personal digital assistants (PDAs) are in use as well. Even the iPods those employees are listening to on their way to and from work can be used to transport files. The location of sensitive data has never been more fluid.

The return on investment for erecting firewalls, virtual private networks (VPNs), proxies, and other perimeter protection devices in order to secure the data that is inside our networks is diminishing. What we as security professionals need is a perimeter that is as mobile as our data. Encryption may be the only security perimeter that is actually capable of moving with that data, and

Continued

provides a strong boundary between the data and the attacker. The TPM can help us build this new mobile perimeter by allowing us to use strong encryption while the cryptographic keys are protected in the TPM.

Trusted Platform Features

According to the TCG, a trusted platform must provide four features: protected capabilities, integrity monitoring, integrity storage, and integrity reporting. A trusted platform maintains certain storage locations, such as platform configuration registers (PCRs) and key storage slots in the TPM, where sensitive data is stored and processed. These locations may be accessed only by protected capabilities, which are the set of commands that are allowed to access and process the data.

The other three features of the trusted platform describe the system's trustworthiness. For instance, we would like to know whether the system has been in a secure state, or whether it has allowed some subject to access objects in an unsupported manner. It is actually acceptable (although undesirable) for the system to enter an insecure state. However, it is unacceptable for the system to lie about the states it has been in. Integrity monitoring is the collection of data on the state of the system, and other metrics which define the trustworthiness of the system for us.

These metrics are used to make judgments concerning the system's trustworthiness, so they must be adequately protected. The storage of integrity reporting measurements in protected locations is termed *integrity storage*. In addition to storing the actual data, integrity storage is used to store hashes of the integrity monitoring data. These hashes serve the same purpose as the hashes available with downloadable files and evidence collected for forensic analysis: They are used to ensure integrity of the data and they serve as proof that a third party has not tampered with the data. The metrics for integrity monitoring can be stored in a logfile, but the digests of the integrity monitoring data are stored in *PCRs*, which are volatile or nonvolatile data storage locations in the TPM itself. *Integrity reporting* is simply the process of vouching for the integrity of the data held in secure storage.

In the next section, you will see how the TCG trusted platform architecture supports and enables these features. It is important to understand how these features allow the trusted platform to start with a very small trust boundary that includes only a portion of the actual system. As the system loads, various pieces of code utilize these features to verify the integrity of other components in the system, and the trust boundary expands to include more and more portions of the system.

Trusted Platform Architecture

According to the TCG, the TPM is the base component of what it has labeled a *trusted platform*, which is a platform that provides users with a higher level of security than the traditional PC. When we discuss the TPM in this sense, we are discussing a somewhat abstract concept that is documented in specification materials developed by the TCG. However, the TPM is now also a part of real PCs.

Although the TCG is still in the process of developing the library of specifications which should be used together to form the trusted platform, we now have the real hardware for the TPM, Microsoft has implemented an operating system that provides a unified interface for the TPM in Windows Vista, applications within Windows Vista take advantage of the TPM, and we will soon see third-party applications that will run on the Windows Vista TSS and will take advantage of the TPM. Therefore, it is important to understand the TPM from two perspectives: We must look at it as a part of the architecture of the trusted platform as conceptualized by the TCG in its document, *TCG Specification Architecture Overview: Specification Revision 1.2*, found at www.trustedcomputinggroup.org/specs/IWG/TCG_1_0_Architecture_Overview.pdf; and we must understand how Microsoft has implemented these specifications by examining the TPM services architecture in Windows Vista.

First, we will look at the TCG trusted platform architecture. It is important to cover this architecture first in order to understand what a trusted platform is and how it works, and to see how the TPM fits into that. Also, Microsoft and computer manufacturers have not simply ripped the TPM from this trusted platform architecture and used it in isolation for some unintended purpose. They are taking the initial steps in trying to implement this evolving design for a secure PC. Once we understand the trusted platform from the perspective of the TCG, we will take a look at the blueprint for Windows Vista TPM services.

The TCG Trusted Platform

The trusted platform needs to have *roots of trust*, which are parts of the system that must be trusted and must enable, at a minimum, the ability to verify the trustworthiness of the rest of the system. Generally, three roots of trust are required: the root of trust for measurement (RTM); the root of trust for storage (RTS); and the root of trust for reporting (RTR). The RTM is actually the normal computing engine for the platform (generally the CPU in the case of a PC) while it is controlled by the core root of trust for measurement (CRTM), which is the set of instructions the computing engine executes in performing its trusted measurement duties. This set of

instructions is supposed to be either a portion of the basic input/output system (BIOS) (the BIOS boot block) or the entire BIOS.

A key concept in discussing the trusted platform is the Trusted Building Block (TBB), which is nothing more than the CRTM and the TPM working together. These two components are physically present on the motherboard, and they must be *immutable*. Every time the system is reset or booted, the CRTM must have first control of the system. In other words, no code may be executed before the CRTM is executed. Following that, the CRTM may not hand off control to another component until it has measured that component to determine its trustworthiness.

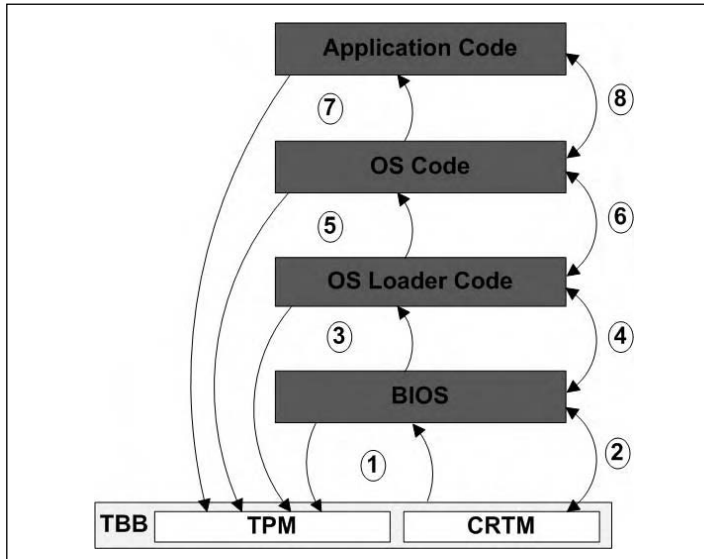
NOTE

An object whose state cannot be modified after it is created is considered *immutable*.

This assumes that trusted mechanisms are being used to boot and run the platform. It is important to remember that the TCG has explicitly stated in its specifications that using the TPM is not mandatory. Users take advantage of the TPM and Windows Vista TPM services on an opt-in basis. This means if the user of the platform wants her system to load just like PCs have loaded in the past—where no trust assurance is provided—she can do that. She can leave the TPM disabled in the BIOS, and not take advantage of TPM functionality.

However, assuming she has enabled the TPM and wants to boot into an operating system on a more trustworthy platform, the boot process starts by loading the CRTM into the CPU and the CRTM begins executing. This must always be the first thing that occurs in a trusted platform following a system reset. All trust in the platform is derived from the TBB, so we have to be absolutely certain that every other piece of code the platform runs comes after the TBB. Otherwise, if a malicious piece of code is inserted into the chain of execution at the beginning, nothing else following it can be trusted.

The system reset loads the CRTM, and it begins executing. At this point, we have a trusted platform. We are at a base state. Only our TBB is functioning, and at this point it is actually the only part of the platform that we trust. All other code that is loaded and executed becomes trusted when the piece of code executing before it takes an integrity measurement of it. No piece of code is allowed to transfer control to another piece of code until it has determined that the code is trustworthy. Once the CRTM is running, we can see the rest of the boot process, as shown in Figure 4.1.

Figure 4.1 The Trusted Platform Loading Process

Here is an explanation of each step shown in Figure 4.1:

1. The CRTM takes integrity measurements of the remaining BIOS code. The measurements may be written to the Stored Measurement Log (SML) or recomputed whenever needed, and a digest of this measurement is created using the SHA-1 hashing algorithm. The SHA-1 digest is written to PCR[0].
2. If the CRTM determines that the measurement it has just taken is evidence of trustworthiness, it passes control of the platform to the remaining BIOS code.
3. The BIOS takes integrity measurements of the platform configuration settings, firmware code, and the code that loads the operating system (OS). The measurements may be written to the SML or recomputed whenever needed, and digests of these measurements are created using the SHA-1 hashing algorithm. The digests are written to the appropriate PCRs.
4. If the BIOS determines that the measurements it has just taken represent evidence of trustworthiness, it passes control of the platform to the remaining OS loader code.
5. Once the OS loader code executes, the next step is to pass control to the remainder of the OS. Before this can be done, the OS loader code must

establish trust in the OS code. Trust measurements are taken against this code, and again, a digest is written to the appropriate PCR in the TPM.

6. If the OS loader trusts the remainder of the OS based, again, on trust measurements, it passes control to the remainder of the OS code.
7. Once the OS is loaded and running, it will control the TPM. However, at times, other applications will need to utilize the TPM. Before the OS can transfer control to an application, it must establish that the application code is trustworthy. It can do this just as it did previously: It takes integrity measurements on the application code and writes a digest to the TPM.
8. Once trust in the application has been established, the OS may transfer control over to it.

You can see by the preceding steps that all trust in the trusted computing platform is based upon the ability to trust the CRTM and TPM. All other trust in the system is derived or induced from this initial TBB. The trusted platform is indeed built upon this relationship between the CRTM and the TPM, so calling them the trusted building blocks is very appropriate. This fact also speaks to the importance of ensuring that the TPM and CRTM are immutable and physically secured to the platform. If we are to rely on the TBB to vouch for the platform's trustworthiness, we must be sure that these two pieces are relatively impervious to modification. From this basis, the boundary of trust is extended to include more parts of the system. However, at any point during loading, if a piece of code does not pass muster during the integrity measurements, control is not transferred to that piece of code and the loading process stops.

Now that we understand how the trusted platform loads, we see the importance of the features of the trusted platform that we touched upon in the preceding section. Measurements are taken (integrity monitoring), they are stored in PCRs (integrity storage), the stored measurements must reliably be reported back to third parties that challenge the system's trustworthiness (integrity reporting), and secure commands (protected capabilities) are used to carry out these operations.

The Physical TPM Interface

Now that we understand the importance of being able to rely upon the physical connection of the TPM to the motherboard as a key assumption for the TBB, it is worth noting the characteristics of that physical connection. The TPM should utilize a Low Pin Count (LPC) bus interface to the motherboard chipset, and a TPM-write cycle and TPM-read cycle should be provided to the TPM. Only the TPM can use these cycles, and the LPC bus must be protected from interference by other devices.

Hardware manufacturers must implement these features. In addition, hardware manufacturers must implement secure firmware for the TPM, provide protection against dictionary attacks (often referred to as *antihammering protections*), and provide other countermeasures designed to combat hardware attacks.

Notes from the Underground...

As We Adapt, So Do the Attackers

As you read and learn about the powerful protection mechanisms that the TPM affords us, it is easy to be lulled into a false sense of security. Although the TPM allows us to avoid some vulnerabilities associated with software-only cryptographic solutions, this does not mean we are completely safe. It simply means that attackers will seek new avenues of attack. Many of these are more difficult and require more skill than what is currently required, though.

For a good briefing on penetration testing the TPM and Windows Vista TPM services, check out Doug Maclver's presentation from Hack in the Box 2006, "Penetration Testing Windows Vista BitLocker Drive Encryption," available at http://packetstormsecurity.org/hitb06/DAY_2_-_Douglas_Maclver_-_Pentesting_BitLocker.pdf.

Binding, Sealing, and Attestation

So far, we have been able to gain an understanding of what a trusted platform is and how it loads. We understand the boot process, and the method by which successive parts of the platform are incorporated into the trust boundary. We can see that each piece of code that controls the platform must not relinquish control to another until it collects metrics on that code and brings it into the trust boundary. All of these are critical functions, yet at the same time many services are available to the user or user mode applications that rely on the TPM. These rely on the capability of the TPM to carry out *binding*, *sealing*, and *attestation* functions.

Binding is essentially the process of encrypting an encryption key. As an example, Microsoft's BitLocker Drive Encryption must use a Volume Master Key (VMK), which encrypts the Full Volume Master Key (FVEK). The SRK, which is created when ownership of the TPM is taken and is stored in the TPM itself, is used to encrypt the VMK. This process is called *binding* or *wrapping*. Through binding or wrapping keys, users can be assured that the keys are secured even though they are

not actually stored in the TPM, simply because the keys cannot be decrypted without a key that is stored in the TPM.

Sealing is the process of binding a key and tying it to certain platform characteristics. So, we know that binding is encrypting a key. Sealing goes one step further and associates the wrapped key to the state of the platform. For instance, it might check the software version, or it might check which versions of dynamic link libraries (DLLs) are loaded, and it will not unwrap the key it has sealed unless the platform has the same DLLs or software versions loaded.

Attestation is just a fancy name for the process of assuring that information is accurate. This is obviously a critical concept for the trusted platform, because as we have seen, all trust in the system is based on taking measurements and then checking those measurements. If the system cannot *attest* to the accuracy of that information, there can be no trust in the platform. An Attestation Identity Key (AIK) is a special 2,048-bit RSA key pair created and stored in the TPM that is used specifically for attestation. Numerous AIKs can be created after ownership of the TPM is taken, and they are used to digitally sign messages proving either the contents of the message or the authenticity of an entity (either a user or a platform). There are four types of attestation as far as the trusted platform is concerned:

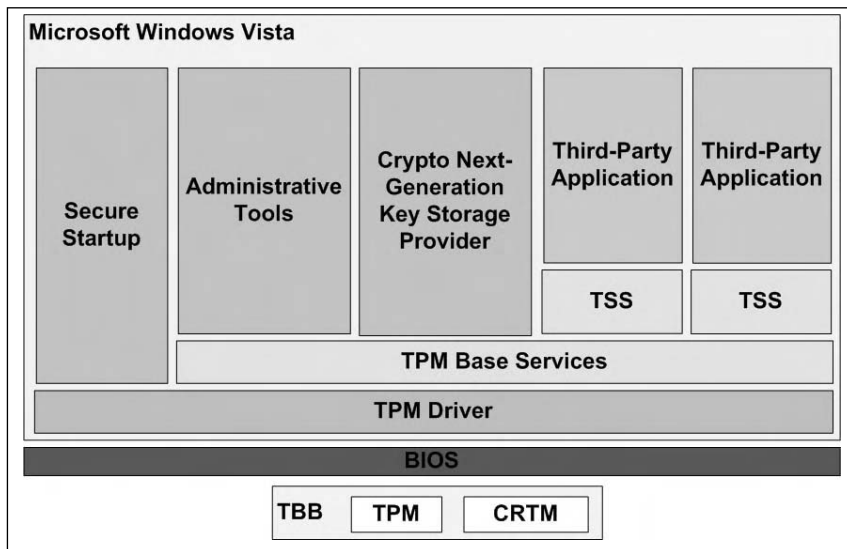
- **Attestation by the TPM** The TPM uses an AIK to digitally sign some data that is held within the TPM itself, proving that the TPM is active and can verify itself.
- **Attestation of the platform** A set of PCRs are signed using an AIK so that the platform can report its integrity to a third party.
- **Attestation of the platform** The platform proves it can be trusted to report its integrity measurements by providing the credentials that were used to create an AIK credential.
- **Authentication of the platform** The platform uses a nonmigratable key, such as an AIK, to authenticate itself. This is strong device authentication.

Your Windows Vista PC

During the booting of a trusted computing platform based on Windows Vista, there is a point where the BIOS must transfer control to Windows Vista. We've covered the basics about what occurs before that transition and how the foundation of the trusted platform is established. We're mainly concerned with what happens after that transition occurs.

Figure 4.2 shows the TBB and BIOS in relation to the Windows Vista architecture. The first thing worth noting is that the TSS sits atop the TPM driver and the TPM Base Services (TBS). The TPM driver is provided by the TPM manufacturer, the TBS is a component of Windows Vista, and the TSS is the implementation of the TCG specification of the TSS. So, the TCG has defined the specs for a component at the bottom of your trusted computing platform, the TPM, as well as a component toward the top, the TSS.

Figure 4.2 The Windows Vista TPM Architecture



There are a few other important notes to consider with regard to the Windows Vista TPM architecture. It should not be surprising, but the TPM driver is required for the functioning of all other parts of the architecture above it. However, unlike most drivers, it is required before the OS loads because the OS loader needs to access the TPM if it is using a secure startup mechanism. Therefore, a TPM driver is also integrated into the BIOS code. Also, Microsoft has created the TBS in order to serve as a mediator between higher-level applications and the TPM driver, which is basically the TPM hardware itself, as far as these applications are concerned. Finally, for all non-Windows applications, the TSS provides a standard API for interacting with the TPM.

Currently, Windows Vista does not take integrity measurements of third-party applications, nor could it. Much like the BIOS, it is incumbent upon the third-party applications to provide measurements during a trusted state. If the applications could provide these metrics to Windows Vista, it can control whether those applications

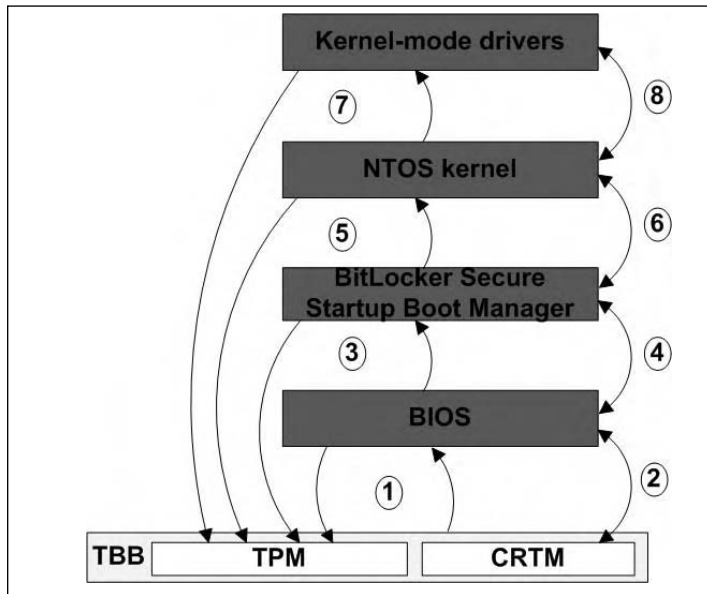
load depending on whether they have been corrupted, resulting in measurements that differ from the “trusted state” measurements. Any updates to these applications that would result in new integrity measurements need to be performed in a controlled manner, where this type of control of the application is shut off momentarily, the application code is updated, and then new measurements are provided.

That being said, the 64-bit version of Windows Vista is providing this level of protection for some pieces of code which it can control. Given that this sort of integrity checking would break most 32-bit applications which were developed with no security mechanisms such as this in place, Code Integrity has been implemented in only the 64-bit version of Windows Vista to prevent breaking already existing 32-bit applications. Code Integrity utilizes the processes and architecture of the trusted platform we covered in the previous sections in order to enable the following features:

- Verification of all drivers that are important to the boot process, including the Hardware Abstraction Layer (HAL) and the OS kernel, by the OS boot loader (which is now Winload)
- Verification of the integrity of all code that executes in a protected process
- Verification of the integrity of all kernel mode drivers
- Verification of the integrity of all user mode binaries that employ cryptographic functions
- Verification of the integrity of all user mode binaries that execute in a protected process used for high-definition media content
- Verification of the integrity of the kernel code itself
- Verification of the integrity of a specific set of user mode binaries

It is not too difficult to see how this Windows Vista architecture relates to the architecture we described when we talked about the more abstract features and functions of the TCG trusted platform. Let’s revisit the trusted platform loading process, but this time we’ll look specifically at our Windows Vista trusted platform (see Figure 4.3).

Figure 4.3 The Windows Vista Using BitLocker Secure Startup Loading Process



Here is an explanation of each step shown in Figure 4.3:

1. The CRTM takes integrity measurements of the remaining BIOS code. The measurements may be written to the SML or recomputed whenever needed, and a digest of this measurement is created using the SHA-1 hashing algorithm. The SHA-1 digest is written to PCR[0].
2. If the CRTM determines that the measurement it has just taken is evidence of trustworthiness, it passes control of the platform to the remaining BIOS code.
3. The BIOS takes integrity measurements of the platform configuration settings, firmware code, and the MBR. The measurements may be written to the SML or recomputed whenever needed, and digests of these measurements are created using the SHA-1 hashing algorithm. The digests are written to the appropriate PCRs. See Table 4.1.
4. If the BIOS determines that the measurements it has just taken represent evidence of trustworthiness, it passes control of the platform to the Windows Vista boot manager, which performs its own integrity checks as part of the secure startup process.

5. Once BitLocker’s startup mechanism takes control, it takes integrity measurements of the New Technology File System (NTFS) boot sector and boot block, and it takes a measurement of the boot manager. These values are stored in PCRs 8 through 10, and PCR 11 is extended with a measurement after the VMK has been unsealed. BitLocker also looks back on the code that has already executed in a sense because the VMK it used for securing the drive has been sealed using PCRs 0, 2, and 4, in addition to 8 through 11.
6. So, once BitLocker’s secure startup process has established trust in the code that preceded it, as well as in the code to which it is about to pass control, it unseals the VMK and decrypts the drive before passing control to the NTOS kernel.
7. Once Windows Vista is loaded and running, it will be responsible for validating the integrity of code that is to run. It does this just as it did previously: It takes integrity measurements of the code, and if it finds an inconsistency when compared to previous measurements, it will not allow the code to load.
8. Once trust in the code is established, Windows Vista allows it to load.

Table 4.1 Interesting PCR Usage Description

| PCR Number | Description of Data Stored in PCR | Entity Writing to This PCR |
|------------|--|----------------------------|
| 0 | An SHA-1 digest of the CRTM version ID, firmware for embedded devices, and remaining BIOS code to which the CRTM is going to transfer control. | CRTM |
| 1 | An SHA-1 digest of the motherboard configuration settings. | BIOS |
| 2 | An SHA-1 digest of the option ROM code. | BIOS |
| 3 | An SHA-1 digest of the option ROM configuration settings. | BIOS |
| 4 | An SHA-1 digest of the Initial Program Loader (IPL) code. This is the MBR. | BIOS |
| 5 | An SHA-1 digest of the IPL (MBR) configuration settings. | BIOS |
| 6 | An SHA-1 digest of state transition and wake events. | BIOS |

Continued

Table 4.1 continued Interesting PCR Usage Description

| PCR Number | Description of Data Stored in PCR | Entity Writing to This PCR |
|------------|--|----------------------------|
| 8 | An SHA-1 digest of the NTFS boot sector. | Windows BitLocker |
| 9 | An SHA-1 digest of the NTFS boot block. | Windows BitLocker |
| 10 | An SHA-1 digest of the boot manager. | Windows BitLocker |
| 11 | An SHA-1 digest used for BitLocker access control. | Windows BitLocker |

The Role of the TBS

The TBS is designed to serve as the scheduler and the TPM resource controller in Windows Vista. The TPM has scarce resources, with only a handful of key slots and session slots in volatile storage, so it is necessary to virtualize these resources and to queue access to them. The TBS accepts commands from higher-level software, performs some validation checks, and then routes the commands down to the TPM through the TPM driver. The TBS may even need to mediate resource requests from multiple instances of the TSS. The process of calling down through the TBS for usage of TPM resources goes like this:

1. A function of the application that requires the TPM is called down through the TSS.
2. The TBS mediates the request and manages the TPM resource.
3. A trust measurement is made of the code that is requesting control.
4. If the measurement reflects a trustworthy piece of code, control is transferred to the application.
5. Once the request is processed, the result is returned to the application.
6. The TBS is then free to process the next job in the queue.

The TBS processes commands based on priority, not on a simple first in, first out stack. When a resource, such as key slots, for example, runs out, the TBS finds the least recently used key slot, saves the key, and then places the new request in the freed slot. If the application or service that had originally requested the slot was evicted, the TBS recognizes this and again makes a least recently used decision in order to free another slot.

One last point of note about TBS is that this component sits at the bottom of the user mode TPM architecture in Windows Vista. At the top of the kernel mode is the TPM driver. The connection between these two components is the conduit between user mode and kernel mode operation as far as TPM is concerned.

Configuring and Managing the TPM on a Stand-Alone System

Configuring the TPM consists of two basic tasks. First, you must *enable* the TPM in the BIOS in order for Windows Vista to even recognize that the chip exists in your system. Second, you must *initialize* the TPM. During the initialization process, you set the TPM to *on*, which is the process by which Windows Vista starts the services provided by the TPM, and you *take ownership* of the chip, which is the process of setting the TPM owner password. The Storage Root Key (SRK) is also created and stored in the TPM at this time. First, we will look at what relevant BIOS settings you need to be worried about, and then we'll cover a few ways in which you can initialize the TPM within Windows.

Tools & Traps...

A TPM Term Quick Reference

As with any technical subject, it is impossible to communicate effectively and to understand the topic without first having a common language with which to communicate. The following is a quick reference of the terms commonly used in relation to TPM hardware and the TPM services within Windows Vista:

- **Enable** This refers to the setting in the computer BIOS where you set the TPM chip to be made available to the operating system by the BIOS.
- **Disable** This refers to the setting in the computer BIOS where you set the TPM chip to be made unavailable to the operating system by the BIOS.
- **Endorsement Key** This is an encryption key that the manufacturer has embedded in the TPM chip. It is composed of a private and public key pair, with the private portion of the key never being exported outside of the TPM chip. This private key can be used to hash data in order to verify that the data can be trusted. The

Continued

recipient of the data can use the public portion of the key to decrypt the data, and the recipient knows that only the TPM could have created the hash of the data. This plays an important part in the integrity monitoring, storage, and reporting functions we discussed earlier.

- **Storage Root Key** The SRK is an encryption key that is created when you take ownership of the TPM, and it is stored in the TPM. When you clear the TPM this is erased, and when you take ownership of an already owned TPM it is replaced by a new SRK. The SRK is used as the master wrapping key. See the “Binding or Wrapping” entry, later in this list.
- **Take ownership** Taking ownership of the TPM is simply the process of setting an owner password. Interestingly, access to the TPM is protected by only one-factor authentication. No username is required for authenticating to the TPM; only this ownership password. Effectively, anyone who knows the owner password is the TPM owner.
- **Initialize** The TPM can be in four states: turned off and unowned; turned on and unowned; turned off and owned; or turned on and owned. When you first boot your Windows Vista system the TPM is turned off and unowned. Initializing the TPM is the process by which you turn on the TPM and take ownership of it.
- **Turn off** Once the TPM is initialized, you may turn off the TPM without losing ownership information. Turning off the TPM simply allows you to stop using Windows Vista TPM services without losing ownership information.
- **Turn on** If the TPM is currently in an off state you may turn the TPM on. Turning on the TPM provides access to Windows Vista TPM services. However, in order to use these services, you must have taken ownership of the TPM. If the TPM is already owned, turning it on allows the TPM services to function properly; if it is not already owned, you must take ownership of the TPM.
- **Clear** When you clear the TPM you reset it back to its factory-default state. It will be off and unowned. This function is available within Windows, and is sometimes also available in the computer BIOS.
- **Binding or wrapping** Encryption keys that the user (or applications and services acting on behalf of the user) creates can be encrypted by the TPM and stored outside of the TPM. This process is called *binding* or *wrapping* the key. Any request for those keys must be made to the TPM, which can unbind/unwrap the key, but

Continued

the private portion of the key is never exposed outside the TPM. The TPM uses the SRK to encrypt user keys.

- **Seal** In addition to simply encrypting keys, the TPM can tie those keys to certain metrics concerning the state of the platform, such as what software is installed. The TPM can decrypt the key only if the state of the platform is the same. When a key is wrapped/bound in this way it is called *sealing the key*.
- **Attestation** A trusted platform must be able to vouch for the accuracy and integrity of data. Attestation is the name for the process of vouching for data.

Configuring BIOS Settings

In order to use the TPM in Windows, as with most onboard hardware devices, you must *enable* it in the BIOS. We have been conducting tests on three different systems that contain a TPM chip, and some contain two BIOS settings that control the TPM. Your system should at least have a BIOS setting allowing you to *enable* or *disable* the TPM. This is a simple on or off function, like a light switch. If you leave this set to the Disable setting, the TPM chip is never made available to Windows. You will not see the chip in Device Manager. This setting must be set to **Enable**.

If you see a second setting concerning the TPM, it probably contains TPM options that can otherwise be controlled within Windows. One of the settings for this second TPM option usually allows you to *clear* the TPM, which erases all of the ownership information and cryptographic keys, including the SRK. It may also allow you to activate and deactivate the TPM. These settings are the equivalent of Turn TPM On and Turn TPM Off in the TPM Management Console. If this set of options is available to you, set the TPM to **Activated** here.

WARNING

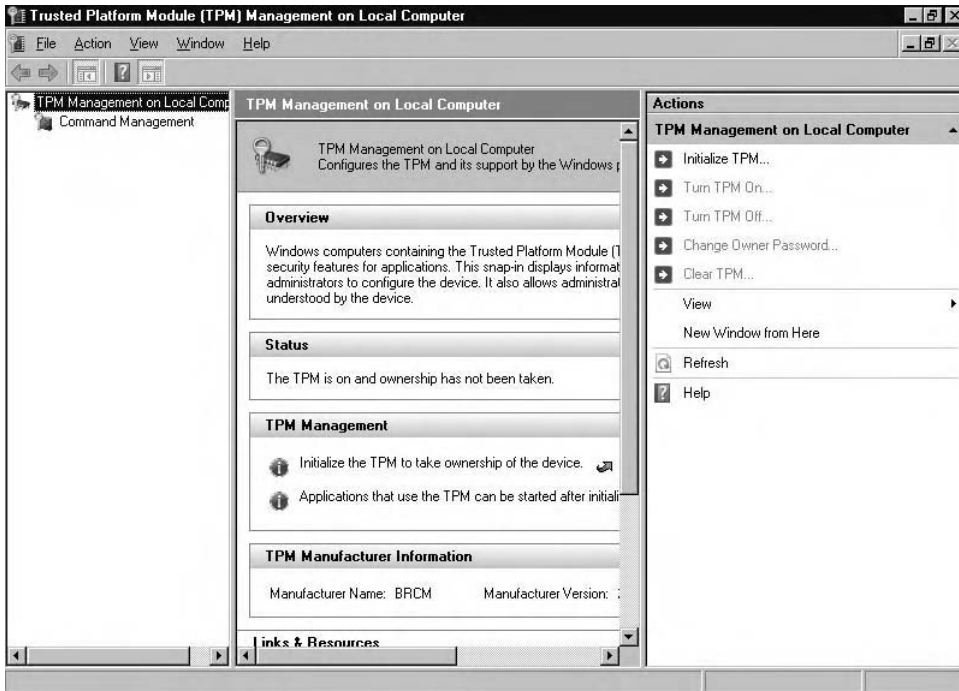
When you are configuring the TPM in the BIOS, in `tpm.msc`, or through scripting, make sure you are careful. For instance, if you used Microsoft BitLocker Drive Encryption to protect a partition and you accidentally clear the TPM, you have just made it very difficult to get your data back. Even if you reinitialize the TPM using the same owner password, it will not contain the same SRK, and it is the SRK that encrypts the keys that were actually used to encrypt your data.

Using the TPM Microsoft Management Console

For anyone familiar with Microsoft Windows computers, the Microsoft Management Console (MMC) should be second nature. It should also be no surprise that Microsoft has built an MMC for configuring and managing the TPM. The consoles continue to be named with the .msc extension, thus invoking the TPM MMC is most handily done by opening the **Run Dialog Box**, typing **tpm.msc**, and then pressing **Enter**.

The TPM MMC has three panes. On the left are just two nodes: the TPM Management on Local Computer node and the Command Management node. When the TPM Management on Local Computer node is selected the center pane displays information specific to the TPM chip in the local computer, and the right pane displays the actions you can perform on the TPM chip, such as clearing the TPM, turning it on or off, and changing ownership of the TPM. Figure 4.4 shows this view of the TPM MMC.

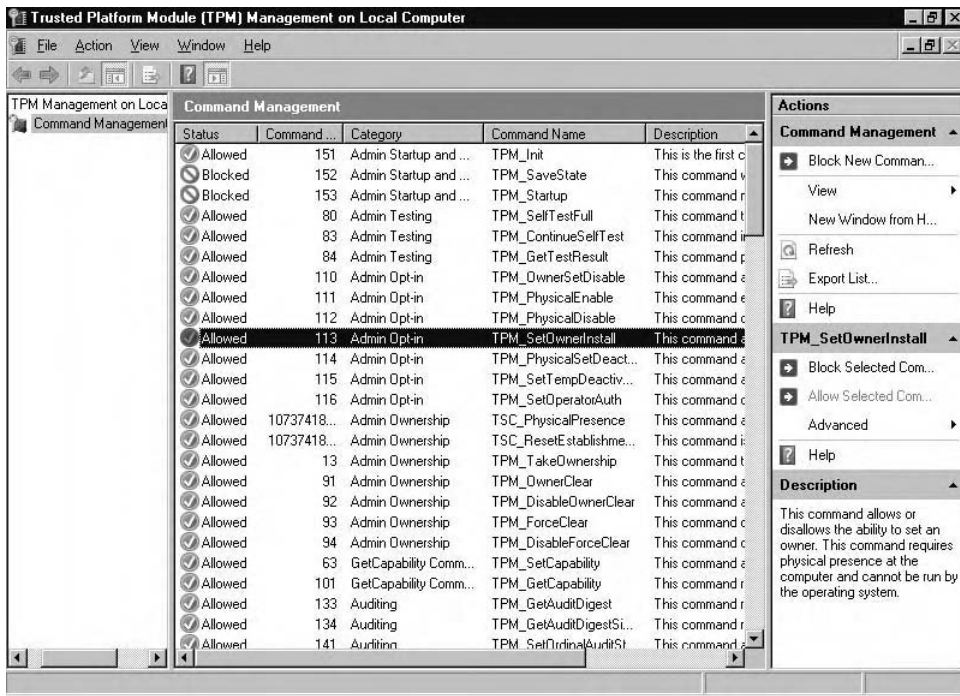
Figure 4.4 The TPM MMC Before Initializing the TPM



The TPM Management on Local Computer node is very straightforward. You can perform five actions on the TPM from this node, and they are shown in the right pane. You can tell the state of the TPM by seeing which actions are grayed out and inaccessible. In Figure 4.4, it is apparent that the TPM is in a factory-default state because the only action available to us is **Initialize TPM**.

When the Command Management node is selected on the left, the full list of TPM commands is displayed in the center pane, and the command-related actions that may be performed are displayed in the right pane. Figure 4.5 shows this view of the TPM MMC.

Figure 4.5 The TPM MMC Command Management Node



Initializing the TPM

The first thing you need to do, before you can take advantage of any TPM-enabled applications, is to initialize the TPM. During the initialization process, you take ownership of the TPM. This means you create an owner password, and a hash of that password is stored in the TPM itself. This owner authorization hash value can also be stored in an XML file, and the credential is needed to perform many of the other TPM management functions, so you want to make sure that it is backed up

somewhere. However, considering that this value is the key to getting into the TPM, it should be protected from disclosure. This means that you should not save it to a Universal Serial Bus (USB) device and then carry that USB device around in the same bag with your laptop.

In order to initialize the TPM, follow these steps:

1. Enter the BIOS and enable the TPM chip.
2. Click **Start | All Programs | Accessories | Run**.
3. Type **tpm.msc** and click **OK**.
4. Click **Continue** to allow the management console to open.
5. The **TPM Management Console** should open, displaying information about the installed TPM chip, as shown in Figure 4.4.
6. On the right side of the **TPM Management Console**, common TPM tasks will be listed, and because the TPM has not been initialized yet, all but the **Initialize TPM** task will be grayed out.
7. Click **Initialize TPM . . .** to start the Initialize the TPM Security Hardware Wizard. The first task is to create a TPM owner password. You will have the option of creating the password yourself, or having the password created automatically (the recommended option).
8. Create the TPM password.

If you select **Automatically create the password (recommended)**, Windows will create a 40-character, all-numeric password. The password will be displayed as shown in Figure 4.6, and you will be required to save the SHA-1 hash of that password to an XML file that has a .tpm extension before the initialization wizard can proceed.

Figure 4.6 Creating the TPM Owner Password Automatically



If you choose the **Manually create the password** option, Windows will ask you to enter a password that is a minimum of eight characters. Using the manual password creation option, you are expected to remember the password (or write it down on a sticky note and attach it to your monitor).

9. Click **Initialize**.
10. After a moment, the wizard will finish and display a message notifying you that it has completed successfully, and that TPM-enabled applications may be used.

Whether you have Windows create the password for you or you create it yourself, you can always store the owner authorization value in a TPM file. Figure 4.7 shows a sample TPM file. Notice that the hash of the owner password is contained between the ownerAuth tags.

Figure 4.7 The Contents of a TPM File

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
This page is a backup of Trusted Platform Module (TPM) owner
authorization information. Upon request, use the authorization information
to prove ownership of the computer's TPM.

IMPORTANT: Please keep this file in a secure location away from your
computer's local hard drive.
-->
<tpmOwnerData version="1.0" softwareAuthor="Microsoft Windows [Version
6.0.6000]" creationDate="2006-12-25T16:56:45-05:00"
creationUser="SYNGRESS\mshepherd" machineName="SYNGRESS">
  <tpmInfo manufacturerId="1112687437"/>
  <ownerAuth>hVOA6LhVpQ8FOZsrKuxyZVgNy5U=</ownerAuth>
</tpmOwnerData>
```



Turning the TPM On

You may turn on the TPM only if it is already in the off state, but you can complete the task regardless of whether you have the password. The following steps lead you through turning the TPM on when you have the password:

1. Click **Start | All Programs | Accessories | Run**.

2. Type **tpm.msc** and click **OK**.
3. Click **Continue** to allow the management console to open.
4. In the **TPM Management Console**, make sure you have the **TPM Management on Local Computer** node selected, and then click **Turn TPM On** in the right pane.
5. When the first screen of the wizard is displayed, select **I have a backup file with the TPM owner password** if you have a TPM file with the password in it, or **I want to type the TPM owner password** if you want to type it in manually.
6. If you opted to type the password manually, the next screen of the wizard requires you to type in the password, and then you can click **Turn TPM On**. If you opted to load the password from a backup file, the next screen will allow you to type or browse to the location of the file, and then you can click **Turn TPM On**.
7. The wizard will turn off the TPM, and then close.

Use these steps in order to turn on the TPM if you have lost the password:

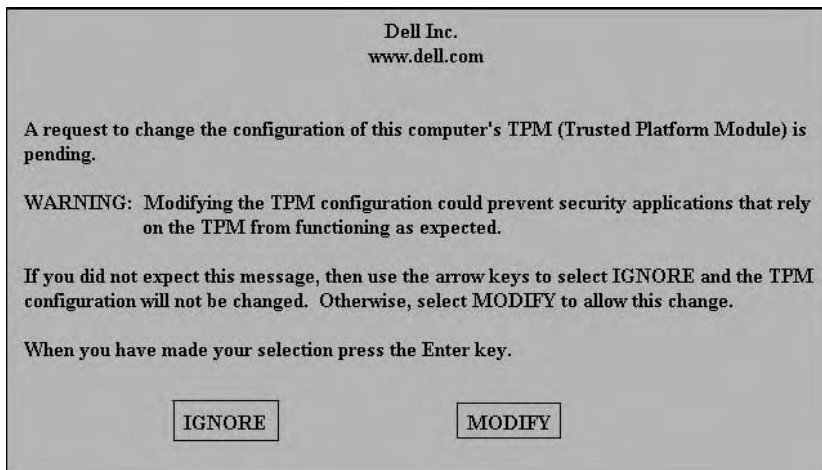
1. Click **Start | All Programs | Accessories | Run**.
2. Type **tpm.msc** and click **OK**.
3. Click **Continue** to allow the management console to open.
4. In the **TPM Management Console**, make sure you have the **TPM Management on Local Computer** node selected, and then click **Turn TPM On** in the right pane.
5. The first screen of the wizard asks you to select whether you have the password in a TPM file, whether you will type it in manually, or whether you do not have the password at all. Select **I do not have the TPM password**.
6. The next screen of the wizard provides some basic instructions, as shown in Figure 4.8. You will need to click the **Restart** button in order to proceed.

Figure 4.8 Turning On the TPM: Final Instructions from the TPM Management Wizard



- When the machine restarts, right after the BIOS loads and before the Windows boot manager has control of the machine, you should see a message telling you that a request to change the TPM configuration has been received (see Figure 4.9). Select **MODIFY** and then press **Enter** to allow the machine to turn the TPM on, and the machine will finish booting.

Figure 4.9 The Computer BIOS Confirmation Message



NOTE

The message you see may be different from the one described in step 7 and shown in Figure 4.8. The BIOS produces this message for the specific platform upon receiving the input provided by the wizard we just ran from within Windows Vista. So, don't expect your Lenovo ThinkPad to display exactly the same message, but it should provide basically the same information and options.



Turning the TPM Off

You may turn off the TPM only if it is already in the on state, but just like turning the TPM on, you can complete the task regardless of whether you have the password. The following steps lead you through turning the TPM off when you have the password:

1. Click **Start | All Programs | Accessories | Run**.
2. Type **tpm.msc** and click **OK**.
3. Click **Continue** to allow the management console to open.
4. In the **TPM Management Console**, make sure you have the **TPM Management on Local Computer** node selected, and then click **Turn TPM Off** in the right pane.
5. When the first screen of the wizard is displayed, select **I have a backup file with the TPM owner password** if you have a TPM file with the password in it, or **I want to type the TPM owner password** if you want to type it in manually.
6. If you opted to type the password manually, the next screen of the wizard requires you to type in the password, and then you can click **Turn TPM Off**. If you opted to load the password from a backup file, the next screen will allow you to type or browse to the location of the file, and then you can click **Turn TPM Off**.
7. The wizard will turn off the TPM, and then close.

Follow these steps in order to turn off the TPM if you have lost the password:

1. Click **Start | All Programs | Accessories | Run**.
2. Type **tpm.msc** and click **OK**.

3. Click **Continue** to allow the management console to open.
4. In the **TPM Management Console**, make sure you have the **TPM Management on Local Computer** node selected, and then click **Turn TPM Off** in the right pane.
5. The first screen of the wizard asks you to select whether you have the password in a TPM file, whether you will type it in manually, or whether you do not have the password at all. Select **I do not have the TPM password**.
6. The next screen of the wizard provides some basic instructions, as shown in Figure 4.10. You will need to click the **Restart** button in order to proceed.

Figure 4.10 Turning Off the TPM: Final Instructions from the TPM Management Wizard



7. When the machine restarts, right after the BIOS loads and before the Windows boot manager has control of the machine, you should see a message telling you that a request to change the TPM configuration has been received (see Figure 4.9 again, and read the note just below the graphic). Select **MODIFY** and then press **Enter** to allow the machine to turn the TPM on, and the machine will finish booting.



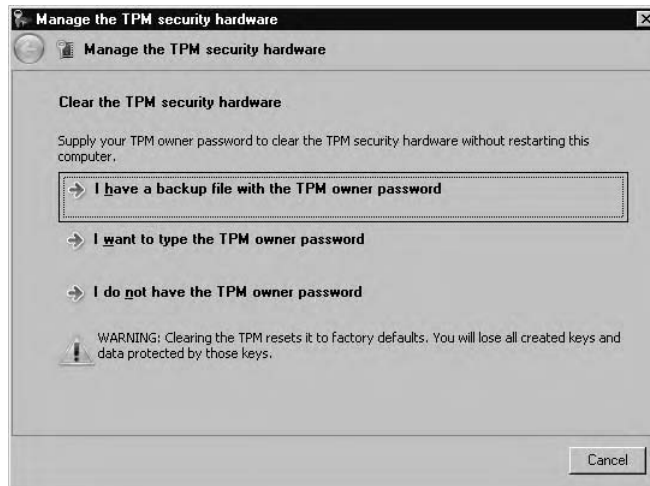
Clearing the TPM

You may clear the TPM only if it has already been initialized (enabled and an owner has been set), but as with the preceding two tasks, you can clear the TPM regardless

of whether you have the password. The following steps lead you through clearing the TPM when you have the password:

1. Click **Start | All Programs | Accessories | Run**.
2. Type **tpm.msc** and click **OK**.
3. Click **Continue** to allow the management console to open.
4. In the **TPM Management Console**, make sure you have the **TPM Management on Local Computer** node selected, and then click **Clear TPM** in the right pane.
5. When the first screen of the wizard is displayed, you will be presented with a justifiably menacing warning message, and options with regard to the TPM owner credentials (see Figure 4.11). Select **I have a backup file with the TPM owner password** if you have a TPM file with the password in it, or **I want to type the TPM owner password** if you want to type it in manually.

Figure 4.11 Clearing the TPM



WARNING

Please take this warning seriously. If you do not have a backup of the keys created or protected by the TPM, any data encrypted with those keys is gone forever!

6. If you opted to type the password manually, the next screen of the wizard requires you to type in the password, and then you can click **Clear TPM**. If you opted to load the password from a backup file, the next screen will allow you to type or browse to the location of the file, and then you can click **Clear TPM**.
7. The wizard will clear the TPM, and then close.

Follow these steps in order to clear the TPM if you have lost the password:

1. Click **Start | All Programs | Accessories | Run**.
2. Type **tpm.msc** and click **OK**.
3. Click **Continue** to allow the management console to open.
4. In the **TPM Management Console**, make sure you have the **TPM Management on Local Computer** node selected, and then click **Clear TPM** in the right pane.
5. The first screen of the wizard asks you to select whether you have the password in a TPM file, whether you will type it in manually, or whether you do not have the password at all. Select **I do not have the TPM password**.
6. The next screen of the wizard provides some basic instructions. You will need to click the **Restart** button in order to proceed.
7. When the machine restarts, right after the BIOS loads and before the Windows boot manager has control of the machine, you should see a message telling you that a request to change the TPM configuration has been received (see Figure 4.9). If you allow the machine to make the change, the TPM will be cleared and the machine will finish booting.

For each of the preceding tasks, you were allowed to perform the task without knowing the owner password. This may seem like a security flaw, but it is not. In order to perform these commands, the following must be true:

- The machine booted successfully, meaning that initial integrity measurements were taken and found to be consistent.
- If the machine uses any secure startup mechanism such as that which comes with BitLocker Drive Encryption, the system was able to verify the integrity of the boot volume, and the TPM successfully unsealed the key used to seal the volume.

- The person logging on was required to present logon credentials to Windows Vista, and these credentials were valid.
- The user that logged on had the appropriate permissions to run the TPM MMC.
- The user was physically present at the system.

Based on these prerequisites, Windows Vista is trusting that it is safe to allow you to turn the TPM on or off or to clear it, even though you do not have the TPM owner credentials. In order for an attacker to use this to his advantage, he would have already had to steal your computer as well as your logon credentials. On top of that, if your computer was protected by a secure startup mechanism, he may have had to steal your PIN, USB storage device, or smart card as well. In addition, the attacker would be able to do nothing more than turn the TPM on, which would basically have no effect on you; turn it off, which would disable any TPM-dependent applications (you could remedy this easily by turning the TPM back on, and these applications would again function properly); or clear the TPM, resulting in a loss of the SRK which is required to access any data that was encrypted using the TPM.

Clearing the TPM has to be considered the worst-case scenario, given that it could lead to a loss of data. It does not provide the attacker with access to your data, though; he already has it by virtue of the fact that he is logged on to Windows. Furthermore, if you (or your organization, if we are discussing a scenario where this is part of an enterprise deployment) are using best practices procedures, you should have the BitLocker Drive Encryption keys backed up to a secure location stored away from the computer, and you can still perform emergency recovery procedures to get the data back.

Also, we should note here that in most cases, the system BIOS provides these functions. So, lacking the protections we mentioned in the five bullets shown earlier, an attacker can just access the BIOS and perform any of these functions much more easily than Windows is allowing him to. Even if you protect your BIOS with a password, this should be considered a much easier attack target than getting into Windows Vista and using the TPM MMC to perform these functions.

These three functions are executed using the *SetPhysicalPresenceRequest* method of the *Win32_Tpm* class, which is described in Table 4.3, later in this chapter. This means that the operation cannot be processed unless the user is present at the platform and authorizes the operation to complete. So, allowing you to turn the TPM on or off without the owner authorization value is not a security flaw. As we shall see in the next section, you are required to provide the owner authorization information in order to change the owner password. This function differs from the previous in that it:

- Provides the attacker with full control of the TPM
- Cannot be performed from outside the OS

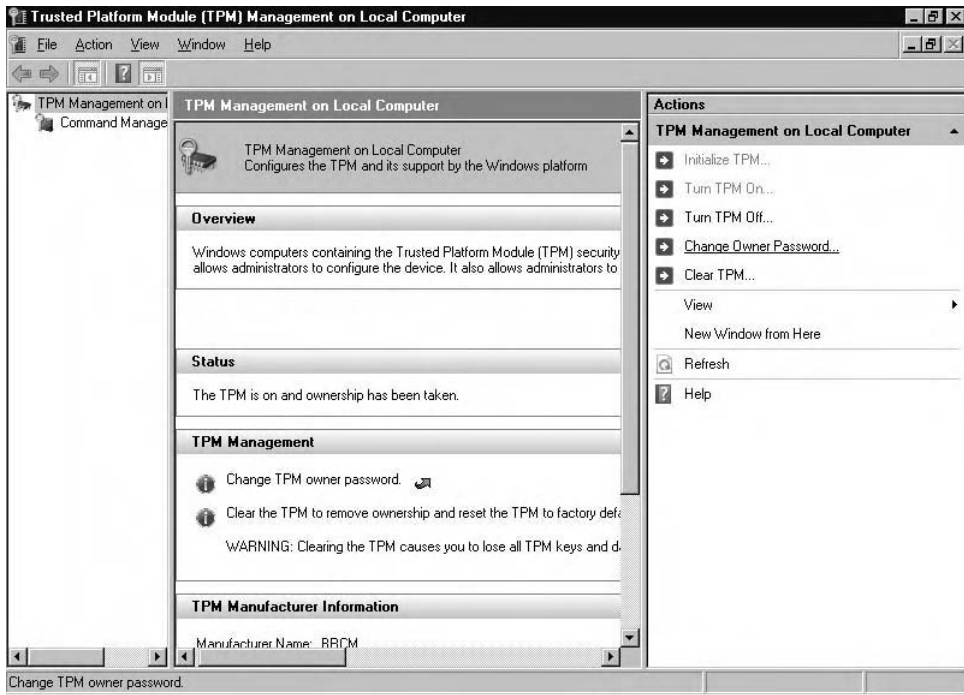


Changing the Owner Password

Follow these steps to change the owner password:

1. Click **Start | All Programs | Accessories | Run**.
2. Type **tpm.msc** and click **OK**.
3. Click **Continue** to allow the management console to open.
4. The **TPM Management Console** should open, displaying information about the installed TPM chip, as shown in Figure 4.12.

Figure 4.12 The TPM MMC with an Owner Set and the TPM Turned On



5. In the **TPM Management Console**, make sure you have the **TPM Management on Local Computer** node selected, and then click **Change Owner Password** in the right pane.

6. When the first screen of the wizard is displayed, select **I have a backup file with the TPM owner password** if you have a TPM file with the password in it, or **I want to type the TPM owner password** if you want to type it in manually.
7. If you opted to type the password manually, the next screen of the wizard requires you to type in the password, and then you can click **Create New Password**. If you opted to load the password from a backup file, the next screen will allow you to type or browse to the location of the file (see Figure 4.13), and then you can click **Create New Password**.

Figure 4.13 Select the Backup File with the TPM Owner Authorization



8. Create the TPM password.

If you select **Automatically create the password (recommended)**, Windows will create a new, 40-character, all-numeric password. The password will be displayed as shown in Figure 4.2, and you will be required to save the SHA-1 hash of that password to a TPM file before the Change Owner Password Wizard can proceed. The **Change Password** button is grayed out until the file is saved.

If you choose the **Manually create the password** option, Windows will ask you to enter a password that is a minimum of eight characters. Using the manual password creation option, you are expected to remember the password, but you can save the authorization value (a hash of that password) to a TPM file if you wish.

9. Once the new owner authorization information has been saved, you can click the **Change Password** button to finish the process (see Figure 4.14).

Figure 4.14 Finishing the Process of Changing the Owner Password Using an Automatically Created Password



Blocking and Allowing Commands

Blocking and allowing commands allows granular control over the secure operation of the TPM within your Windows Vista system. Currently, 125 commands are listed in the TPM MMC, and they are organized into 26 categories of functionality. Each is represented by an ordinal value, which is the value you will pass to the block and unblock functions.

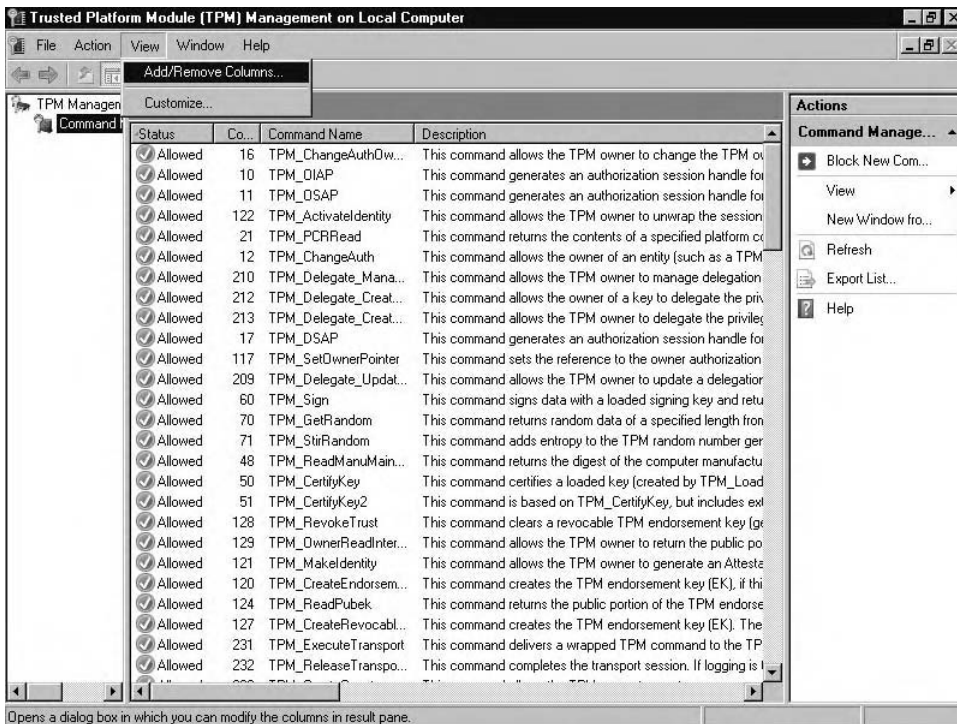
There are three possible lists of blocked commands: the default list provided with Windows Vista; a list maintained on the local machine and managed by local administrators; and the list of commands controlled by Group Policy Objects (GPOs). If a TPM command exists in any of the lists, it will be blocked by the entity that controls command execution, the TBS. The TBS checks all three lists before passing the attempted command down to the TPM driver for processing. If the command exists in one of the lists, the TBS prevents the execution and returns an error to the service or application that sent the command.

The default list is evidenced by the fact that on a fresh Windows Vista install, some commands are already listed as blocked in the TPM MMC. In Figure 4.4, you can see that commands 152 (*TPM_SaveState*) and 153 (*TPM_Startup*) are blocked, but no administration of the command blocking list has occurred yet. Those are blocked by default.

There is more than meets the eye when it comes to the Command Management node in the TPM MMC. When you are managing the TPM on a system and you have these three lists to worry about, it can be difficult to figure out why a command is being blocked or allowed. Perhaps you've set the default block list to ignore (not recommended, by the way), but one command that you need to be able to use is still blocked. Instead of hunting down the lists in various places, you can use the TPM MMC to display which block list a command is on. To do that, follow these steps:

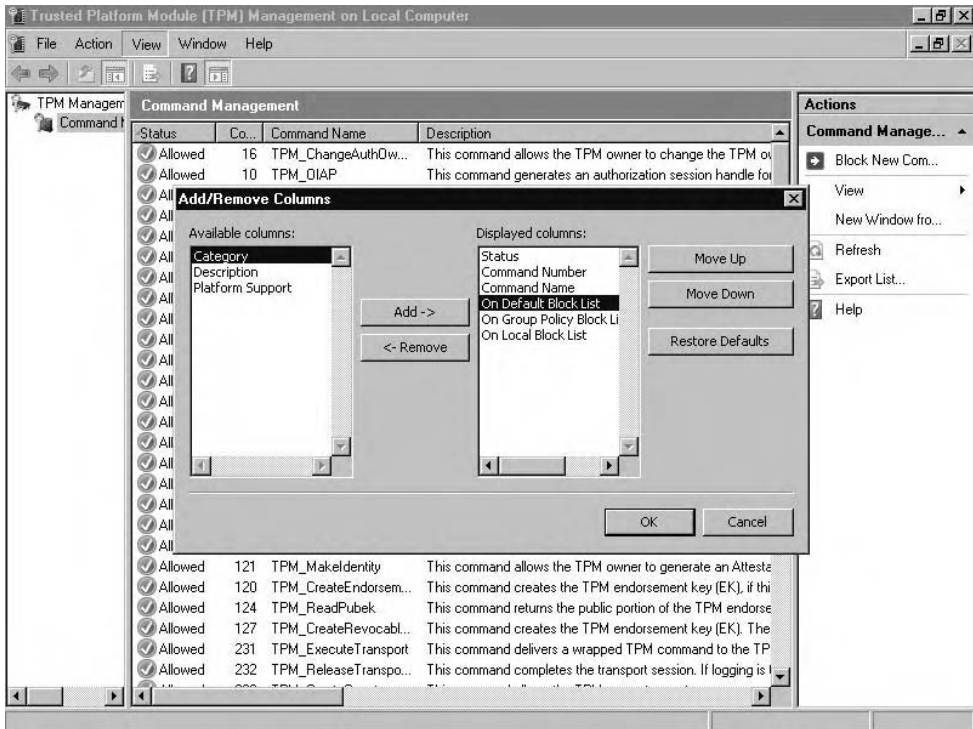
1. Click **Start | All Programs | Accessories | Run**.
2. Type **tpm.msc** and click **OK**.
3. Click **Continue** to allow the TPM MMC to open.
4. Select **Command Management** in the left pane.
5. Click **View | Add/Remove Columns**, as shown in Figure 4.15.

Figure 4.15 Changing the Information Displayed in the TPM MMC



- Use the **Add/Remove Columns** dialog box to add the On Default Block List, On Group Policy Block List, and On Local Block List columns, and remove any unwanted columns (here we have removed Category and Description), as shown in Figure 4.16.

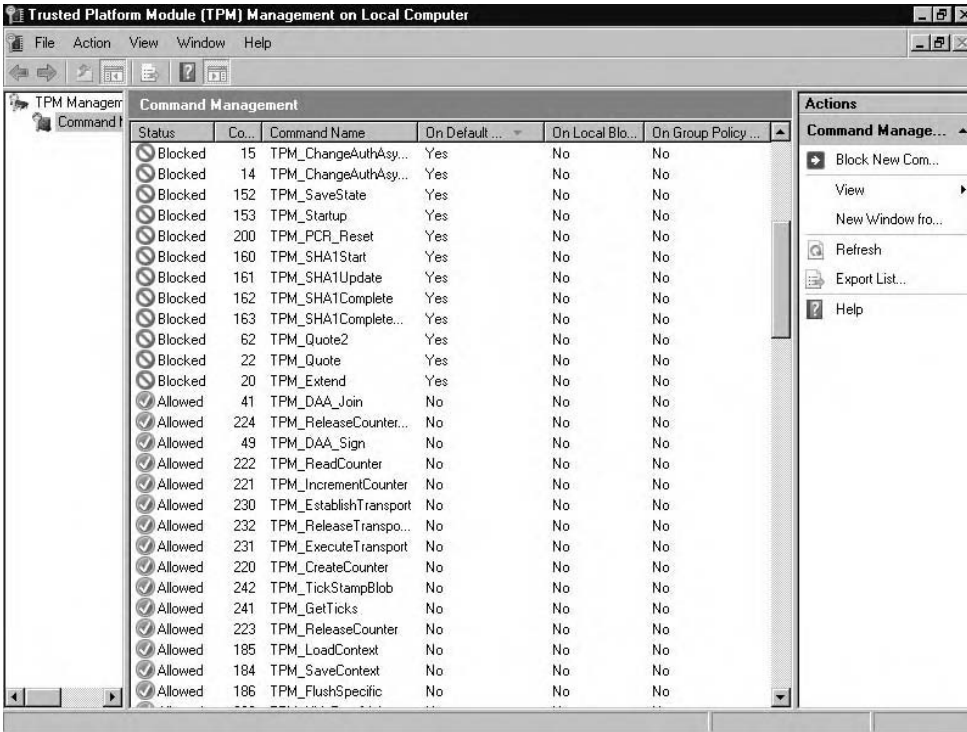
Figure 4.16 The Add/Remove Columns Dialog Box for the Command Management Node



- Click **OK** and the Command Management node will appear as it does in Figure 4.17.

Now that we can actually see what commands are on which lists, let's take a look at how you manage them. Commands that are on the default block list cannot be allowed through the TPM MMC unless you set the Local Computer Policy to ignore the default list. We would caution at this point that in a stand-alone system, the default block list should always be used. The commands are on the default list because they are either deprecated or can compromise privacy.

Figure 4.17 The TPM MMC Command Management Node Displaying the Block List Columns



The Local Computer Policy contains four settings that are relevant to the TPM. You can find these settings in the *Local Computer Policy\Computer Configuration\Administrative Templates\System\Trusted Platform Module Services* node, and the settings are described in Table 4.2. You can open the Local Computer Policy using the following steps:

1. Click **Start | All Programs | Accessories | Run**.
2. Type **gpedit.msc** and click **OK**.
3. Click **Continue** to allow the management console to open.

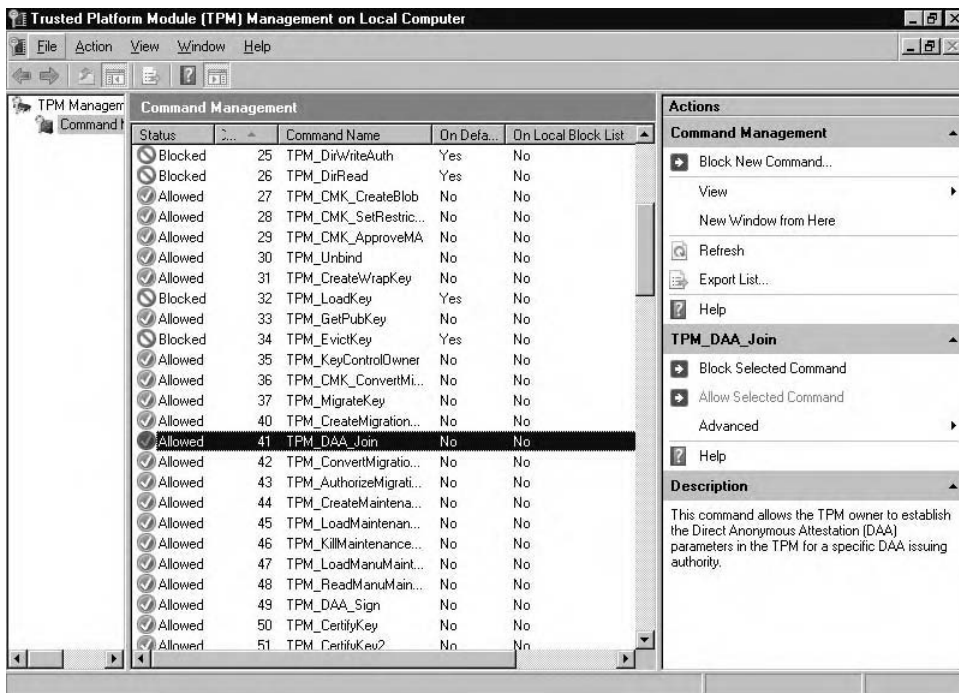
Table 4.2 TPM Group Policy Settings for Stand-Alone Systems

| Setting Name | Description | Possible Configurations | Result |
|--|--|---------------------------------------|--|
| Turn on TPM backup to Active Directory Domain Services | This setting controls whether the hash of the TPM owner password is backed up to Active Directory. | Not Configured Disabled Enabled | TPM owner credential is not backed up to Active Directory. TPM owner information is silently backed up to Active Directory when the owner credential is changed. TPM owner information is silently backed up to Active Directory when the owner credential is changed; if the backup fails, the owner credential is not changed in the TPM. |
| Configure the list of blocked TPM commands | This is the Group Policy block list. You may enable it and add blocked commands. | Not Configured Disabled Enabled | The Group Policy block list is not used. The Group Policy block list is used, and any commands added to the list using the Show Add function of this setting will be blocked. |
| Ignore the default list of blocked TPM commands | Controls whether the default list of blocked TPM commands is used. | Not Configured Disabled Enabled | The default list is used to block TPM commands. The default list is ignored, and commands on this list are allowed. |
| Ignore the local list of blocked TPM commands | Controls whether the local list of blocked TPM commands is used. | Not Configured Disabled Enabled | The local list is used to block TPM commands. The local list is ignored, and commands on this list are allowed. |

Because we are managing the TPM on a stand-alone system, we think the best practice for managing the TPM is to leave the default block list in place, and to use the local list to add other commands we may want to block. There is no reason to add another layer of confusion to the scenario by utilizing both the Group Policy list and the local list to set our own blocked commands. Also, because we are managing a stand-alone system, we do not have an Active Directory infrastructure to which we can back up the owner authorization credentials. So, basically the best method for managing the TPM is to leave all the Local Computer Policy settings in their default states, and to utilize only the TPM MMC to block and allow commands. Here is the easiest method for blocking/allowing commands within the MMC (these directions assume you already have the TPM MMC opened to the Command Management node):

1. Select the command you want to block/allow, as shown in Figure 4.18.

Figure 4.18 The TPM MMC Command Management Node before Blocking a Command

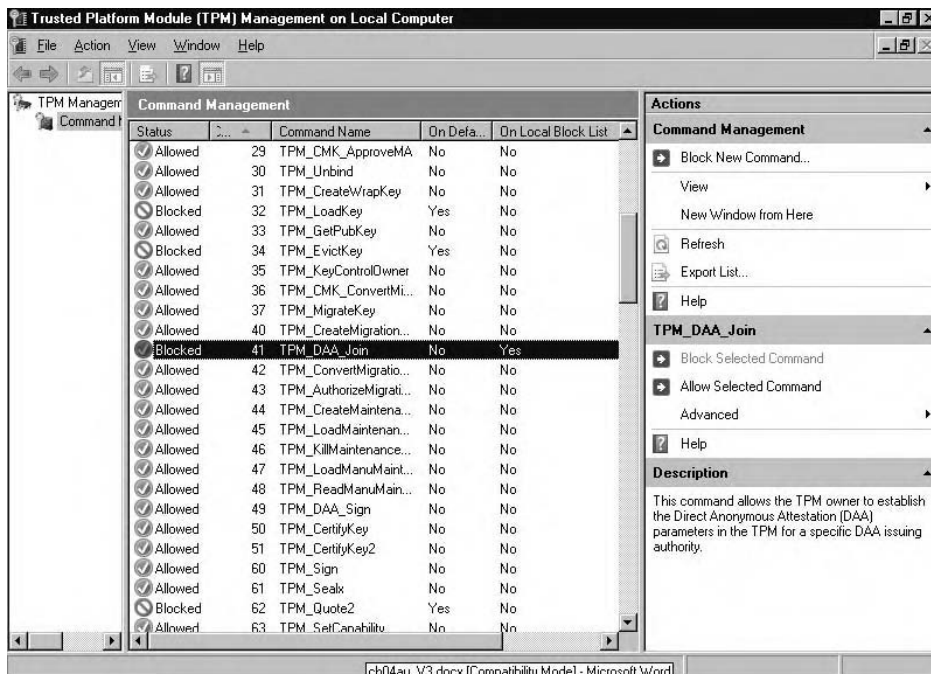


2. If the command is currently blocked and you want to allow it, click **Allow Selected Command**. If the command is currently allowed and you want to block it, click **Block Selected Command**.
3. If you blocked the command, the MMC will add the command to the local block list. If you allowed the command, the MMC will remove the command from the local block list. The new status will immediately be reflected in the MMC, as shown in Figure 4.19 (make note of the Status and On Local Block List columns for command 41).

NOTE

You'll notice that in Figure 4.19, the command status displays as Blocked, but the icon shown is still the Allowed icon. If you click **Refresh** in the right pane, the icon is updated. Consider this a very minor bug in this prerelease of Windows Vista.

Figure 4.19 The TPM MMC Command Management Node after Blocking a Command

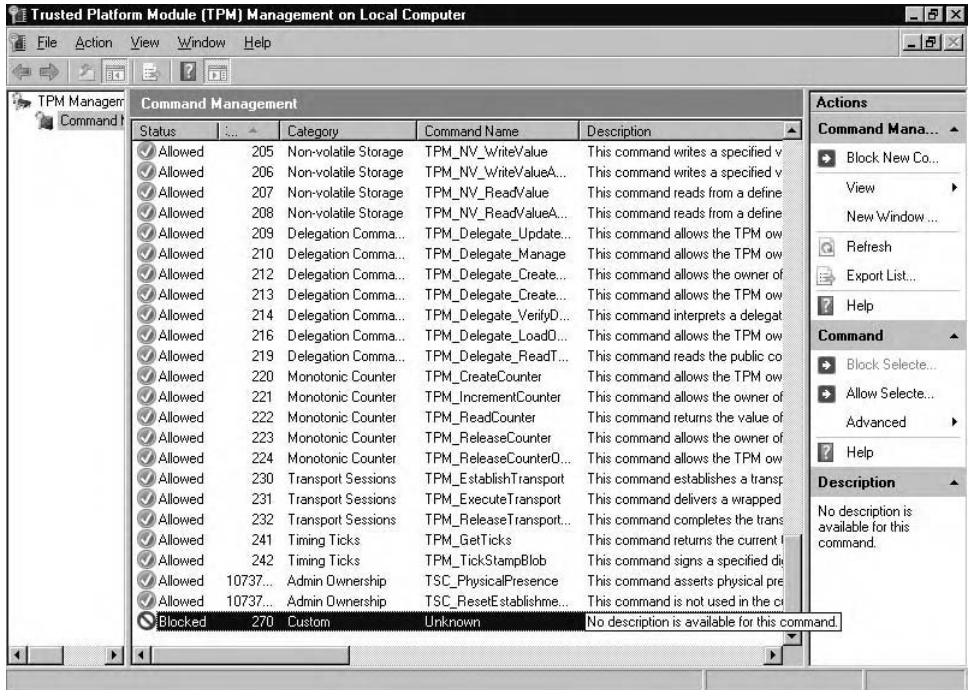


NOTE

If you enjoy doing things the hard way and need a way to really screw something up on your machine, you should know that the default block list, Group Policy block list, and local block list are stored in the Registry. You could avoid the ease of use the TPM MMC provides and edit the Registry, and risk deleting some critical Registry key in the process. You can find the default block list in *HKEY_LOCAL_MACHINE\Software\Microsoft\Tpm\BlockedCommands\List*. You can find the local block list at *HKEY_LOCAL_MACHINE\Software\SYSTEM\ControlSet001\SharedAccess\Parameters\Tpm\BlockedCommands\List*. The Group Policy block list is stored in *HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Tpm\BlockedCommands\List*.

You have probably noticed that a Block New Command function is provided in the right pane as well. This does not provide the same functionality as the Block Selected Command function we just covered, and in fact, you cannot use it to block any of the commands listed in the TPM MMC at this point. All TPM commands agreed upon by the TCG at the time Vista is released are listed in the MMC, and you should block them as described earlier. However, hardware vendors may decide to implement commands that are not a part of the TPM specification, or the TCG may add new commands to the specification. Microsoft implemented the Block New Command function to provide coverage for both of these situations. In order to test the function, just pass it a number that does not correspond to a command that appears in the MMC, and the new command number will appear in the list. In Figure 4.20, you can see that we have blocked command 270.

Figure 4.20 Blocking New TPM Commands



Configuring and Managing the TPM in an Enterprise Environment

The TPM MMC provides a familiar interface for Windows system administrators to manage the TPM on a local machine. However, you cannot use the MMC to perform remote TPM management, and it does not allow for the batch processing of management tasks across multiple machines at once. In the enterprise environment, you're going to need to deploy numerous systems all at once instead of just one. Even in the enterprise where you will not be utilizing the TPM and the services Microsoft has built on it, it is still a good idea to lock users out of being able to manage the TPM locally. Otherwise, your help desk phones are likely to start ringing with calls from users who read about the cool new TPM features somewhere, tried to enable BitLocker Drive Encryption themselves, and in the end lost all the data on their hard drives when they cleared the TPM. Regardless of whether you plan to use the TPM right away, it is now a resource in your organization that needs to be managed.

This presents a whole new slew of challenges, and you know you are not going to be able to enjoy your weekends if you have to manually configure and manage the TPM on all these devices manually. You absolutely do not want to be using the TPM MMC to manage all your devices, and you don't want to be saving the hash of each owner password to one USB key which you'll then toss into your bottom desk drawer. Microsoft has thought of these challenges and has provided some pretty handy deployment tools. Again, these tools should be old news to any seasoned Microsoft systems administrator. Windows Management Instrumentation (WMI) scripting will allow administrators to complete tasks, such as taking ownership of the TPM, turning on the TPM, or turning off the TPM, on multiple systems at the same time. GPOs available with Windows Server 2007 (codenamed Longhorn) provide administrators with a cadre of configuration options that will work with the scripts they use to deploy TPM-based systems.

Tools & Traps...

TPM Management Best Practices

You should use the following best practices to help ensure a secure and manageable deployment of TPM hardware and Windows Vista services within your enterprise:

- First, you need to ensure that you are purchasing systems with TPM chips that comply with the TCG TPM version 1.2 specification.
- Second, make sure that any applications you plan to use comply with version 1.2 of the TCG TSS specification, and have been developed to work with TBS.
- Use WMI scripts to initialize the TPM on multiple systems, if possible, before you deploy hardware to users.
- Use WMI scripts to take ownership of the TPM if you need to do it remotely, or if you are performing the task on multiple systems.
- Store TPM-related data in Active Directory.
- Use unique owner passwords on all systems in the enterprise.
- Never give TPM owner passwords or authorization information to system users.
- If system users need to perform tasks that require TPM owner credentials, it is best to delegate permissions to the users to perform

Continued

the tasks. This will depend on whether the software supports delegation of duties.

- Maintain the list of blocked TPM commands using restrictive policies.
- Do not store the owner password or authorization information on local system media.

Using GPOs and Active Directory

The next version of Windows Server, codenamed “Longhorn,” will include Active Directory schema extensions that support both TPM management and BitLocker Drive Encryption management. (The latter is covered in more detail in Chapter 5). The main thing you need Active Directory to do for you is provide a place to store the hashes of TPM owner passwords so that each TPM may have a unique owner password, and you can store the hashes in a central, secure location. The same goes for the BitLocker Drive Encryption keys. You can store those keys in Active Directory in the event you need to enact emergency recovery procedures. The Active Directory attribute where the TPM owner authorization value is stored is *ms-TPM-OwnerInformation*. Keep this in mind for when you need to run a script, and you need to have the computer’s owner authorization information to call the *Win32_Tpm* method you want.

Preparing Your Pre-Longhorn Domain Controllers

These schema extensions come with Windows Server 2007 by default, but if you are running a domain that utilizes Windows Server 2003 and/or Windows 2000 Server domain controllers, you do not have the necessary objects and attributes in Active Directory to take advantage of these central management and key storage features. In order to extend your Active Directory schema, all of your domain controllers must be running Windows Server 2003 Service Pack (SP) 1 or later because the Active Directory schema version in this release contains the ability to set certain attributes in Active Directory as confidential, which protects them from being read by unauthorized personnel. The TPM and BitLocker Drive Encryption keys should be stored in confidential attributes for obvious reasons.

There is a version of the Microsoft schema preparation utility, *adprep*, which comes on the Windows Vista and Windows Server 2007 DVDs in *sources\adprep*. This

is not a new utility, and anyone who has been through the Active Directory upgrade process should be familiar with it. However, the official word from Microsoft is that this should not be used. This folder is included for informational purposes only, to demonstrate what schema changes will take place. The scripts did work for us on a Windows Server 2003 for Small Business Server SP1 system, but because you are not supposed to be using them at this time, you'll have to run them at your own risk. Once Windows Server 2007 is actually released, official support for using these is sure to come with it.

It is suggested that you make sure you use the latest version of adprep anytime you perform an upgrade of Active Directory. You'll need to be able to access the entire adprep folder (adprep needs those .ldf files too) from your Windows Server 2003 SP1 flexible single master operations (FSMO) server in order to perform the required schema upgrade. When the folder is available, execute the following command from a command prompt (the location of the files is assumed to be *C:\adprep* in the following example):

```
C:\adprep>adprep /forestPrep
```

After a lengthy warning from the utility, if you choose to continue you should see something such as the following:

```
Opened Connection to DC1
SSPI Bind Succeeded
Current Schema Version is 30
Upgrading schema to version 39
Connecting to "DC1"
Logging in as current user using SSPI
Importing directory from file "C:\WINDOWS\system32\sch31.ldf"
Loading entries . . . . .
139 entries modified successfully.
```

```
The command has completed successfully
Connecting to "DC1"
Logging in as current user using SSPI
Importing directory from file "C:\WINDOWS\system32\sch32.ldf"
Loading entries . . . . .
18 entries modified successfully.
```

```
The command has completed successfully
Connecting to "DC1"
Logging in as current user using SSPI
```

```
Importing directory from file "C:\WINDOWS\system32\sch33.ldf"
Loading entries . . . . .
17 entries modified successfully.
```

This continues until you've reached version 39 of the schema. After this, just run:

```
C:\adprep>adprep /domainPrep
```

At this point, you will have an Active Directory schema capable of accommodating the new features of Windows Vista. The downside is that you will not have the proper administrative templates installed to see the corresponding Group Policy settings. However, you can add the following text to your System administrative template, stored in %systemroot%\SYSVOL\Policies\<varies>\Adm\system.adm, in the section where the policy categories are defined, in order to have the same template available as Windows Server 2007 users will have:

```
CATEGORY !!TrustedPlatformModuleServices
    #if version >= 4
        EXPLAIN !!TrustedPlatformModuleServices_Help
    #endif
POLICY !!ActiveDirectoryBackup
    #if version >= 4
        SUPPORTED !!SUPPORTED_WindowsVista
    #endif
    EXPLAIN !!ActiveDirectoryBackup_Help
    KEYNAME "Software\Policies\Microsoft\TPM"
    VALUENAME "ActiveDirectoryBackup"
    VALUEON NUMERIC 1
    VALUEOFF NUMERIC 0
    PART !!RequireActiveDirectoryBackup_Name CHECKBOX
        VALUENAME "RequireActiveDirectoryBackup"
        DEFCHECKED
        VALUEON NUMERIC 1
        VALUEOFF NUMERIC 0
    END PART
END POLICY
POLICY !!BlockedCommandsList
    #if version >= 4
        SUPPORTED !!SUPPORTED_WindowsVista
    #endif
```

```

    EXPLAIN !!BlockedCommandsList_Help
    KEYNAME "SOFTWARE\Policies\Microsoft\Tpm\BlockedCommands"
    VALUENAME "Enabled"
    VALUEON NUMERIC 1
    VALUEOFF NUMERIC 0
    PART !!BlockedCommandsList_Ordinals2 LISTBOX
        KEYNAME
"SOFTWARE\Policies\Microsoft\Tpm\BlockedCommands\List"
        END PART
    END POLICY
POLICY !!IgnoreDefaultList
    #if version >= 4
        SUPPORTED !!SUPPORTED_WindowsVista
    #endif
    EXPLAIN !!IgnoreDefaultList_Help
    KEYNAME "Software\Policies\Microsoft\TPM\BlockedCommands"
    VALUENAME "IgnoreDefaultList"
    VALUEON NUMERIC 1
    VALUEOFF NUMERIC 0
    END POLICY
POLICY !!IgnoreLocalList
    #if version >= 4
        SUPPORTED !!SUPPORTED_WindowsVista
    #endif
    EXPLAIN !!IgnoreLocalList_Help
    KEYNAME "Software\Policies\Microsoft\TPM\BlockedCommands"
    VALUENAME "IgnoreLocalList"
    VALUEON NUMERIC 1
    VALUEOFF NUMERIC 0
    END POLICY
END CATEGORY ; TrustedPlatformModuleServices

```

In the *[strings]* section at the bottom of the file, include the following text:

```

ActiveDirectoryBackup_Help="This policy setting allows you to manage the
Active Directory Domain Services (AD DS) backup of Trusted Platform Module
(TPM) owner information. \n\nTPM owner information includes a cryptographic
hash of the TPM owner password. Certain TPM commands can only be run by the
TPM owner. This hash authorizes the TPM to run these commands. \n\nIf you
enable this policy setting, TPM owner information will be automatically and
silently backed up to AD DS when you use Windows to set or change a TPM
owner password. \n\nIf you select the option to "Require TPM backup to AD

```

DS", a TPM owner password cannot be set or changed unless the computer is connected to the domain and the AD DS backup succeeds. This option is selected by default to help ensure that TPM owner information is available. Otherwise, AD DS backup is attempted but network or other backup failures do not impact TPM management. Backup is not automatically retried and the TPM owner information may not have been stored in AD DS during BitLocker setup.

\n\nIf you disable or do not configure this policy setting, TPM owner information will not be backed up to AD DS. \n\nNote: You must first set up appropriate schema extensions and access control settings on the domain before AD DS backup can succeed. Consult online documentation for more information about setting up Active Directory Domain Services for TPM. \n\nNote: The TPM cannot be used to provide enhanced security features for BitLocker Drive Encryption and other applications without first setting an owner. To take ownership of the TPM with an owner password, run "tpm.msc" and select the action to "Initialize TPM". \n\nNote: If the TPM owner information is lost or is not available, limited TPM management is possible by running "tpm.msc" on the local computer."

BlockedCommandsList_Help="This policy setting allows you to manage the Group Policy list of Trusted Platform Module (TPM) commands blocked by Windows. \n\nIf you enable this policy setting, Windows will block the specified commands from being sent to the TPM on the computer. TPM commands are referenced by a command number. For example, command number 129 is TPM_OwnerReadInternalPub, and command number 170 is TPM_FieldUpgrade. To find the command number associated with each TPM command, run "tpm.msc" and navigate to the "Command Management" section. \n\nIf you disable or do not configure this policy setting, only those TPM commands specified through the default or local lists may be blocked by Windows. The default list of blocked TPM commands is pre-configured by Windows. You can view the default list by running "tpm.msc", navigating to the "Command Management" section, and making visible the "On Default Block List" column. The local list of blocked TPM commands is configured outside of Group Policy by running "tpm.msc" or through scripting against the Win32_Tpm interface. See related policy settings to enforce or ignore the default and local lists of blocked TPM commands."

ActiveDirectoryBackup="Turn on TPM backup to Active Directory Domain Services"

BlockedCommandsList_Name="Configure the list of blocked TPM commands"

IgnoreDefaultList_Help="This policy setting allows you to enforce or ignore the computer's default list of blocked Trusted Platform Module (TPM) commands. \n\nIf you enable this policy setting, Windows will ignore the computer's default list of blocked TPM commands and will only block those TPM commands specified by Group Policy or the local list. \n\nThe default list of blocked TPM commands is pre-configured by Windows. You can view the default list by running "tpm.msc", navigating to the "Command Management" section, and making visible the "On Default Block List" column. The local list of blocked TPM commands is configured outside of Group Policy by running "tpm.msc" or through scripting against the Win32_Tpm interface. See the related policy setting to configure the Group Policy list of blocked TPM

commands. \n\nIf you disable or do not configure this policy setting, Windows will block the TPM commands in the default list, in addition to commands in the Group Policy and local lists of blocked TPM commands."

IgnoreDefaultList="Ignore the default list of blocked TPM commands"

IgnoreLocalList_Help="This policy setting allows you to enforce or ignore the computer's local list of blocked Trusted Platform Module (TPM) commands.

\n\nIf you enable this policy setting, Windows will ignore the computer's local list of blocked TPM commands and will only block those TPM commands specified by Group Policy or the default list. \n\nThe local list of blocked TPM commands is configured outside of Group Policy by running "tpm.msc" or through scripting against the Win32_Tpm interface. The default list of blocked TPM commands is pre-configured by Windows. See the related policy setting to configure the Group Policy list of blocked TPM commands. \n\nIf you disable or do not configure this policy setting, Windows will block the TPM commands found in the local list, in addition to commands in the Group Policy and default lists of blocked TPM commands."

IgnoreLocalList="Ignore the local list of blocked TPM commands"

TrustedPlatformModuleServices="Trusted Platform Module Services"

TrustedPlatformModuleServices_Help="The Trusted Platform Module is a microchip that supports trusted computing services in Windows Vista. These settings control the functioning of the device."

BlockedCommandsList="Configure the list of blocked TPM commands"

BlockedCommandsList_Ordinals2="The list of blocked TPM commands:"

RequireActiveDirectoryBackup_Name="Require TPM backup to AD DS"

SUPPORTED_WindowsVista="At least Microsoft Windows Vista"

Now your Windows Server 2003 SP1 or later domain controller will provide the same functionality as a Windows Server 2007 domain controller does with regard to TPM management. It should be noted that you can use the Group Policy Object Editor (GPOE) or Group Policy Management Console (GPMC) that comes with Windows Vista and Windows Server codenamed "Longhorn" to manage your Windows Server 2003 environment.

Preparing Your Longhorn Domain Controllers

The process of preparing your Windows Server 2007 domain controllers in an all-Longhorn environment is much simpler. There is no need to upgrade the Active Directory schema. The only things missing from these domain controllers are the administrative templates that display the relevant Group Policy settings in the Group Policy Management MMC. Actually, they're not missing entirely. They are installed in the `%systemroot%\PolicyDefinitions` folder on both Windows Vista and Windows Server 2007 systems where the Local Computer Policy reads them from.

All we need to do is copy them to the central store which is part of *SYSVOL* so that they are replicated to all domain controllers and are available for domain GPOs. We need to make sure we copy both the administrative templates and the language-specific files. For English, execute the following commands from a command prompt:

```
C:\>xcopy C:\WINDOWS\PolicyDefinitions\*
C:\WINDOWS\SYSVOL\domain\policies\PolicyDefinitions\
C:\>xcopy C:\WINDOWS\PolicyDefinitions\EN-US\*
C:\WINDOWS\SYSVOL\domain\policies\PolicyDefinitions\EN-US\
```

When the files have been copied, you may need to wait for replication to distribute this change throughout your network. However, on the domain controller on which you just performed the copy, you can start using the Group Policy Object Editor to create a GPO right away.

Blocking Commands

In order to block TPM commands, you don't actually need to meet the requirements set forth in the section on preparing Windows Server 2003 domain controllers. The GPO used to block commands simply pushes changes to the Registry on domain computers. The schema upgrades we covered are necessary only for storing the TPM owner authorization hash and other cryptographic keys related to the TPM (such as BitLocker Drive Encryption keys) in Active Directory. Technically, you could make the changes to your administrative template that were shown on any domain controller in an enterprise running Active Directory, and you'd still be able to utilize the settings related to command blocking. Just be absolutely sure you don't enable the **Turn on TPM backup to Active Directory Domain Services** setting. If you do this, you will require the computer to back up the TPM owner authorization values to Active Directory, but there will be no place for the keys to be stored in Active Directory. The result is that no attempts to set ownership on any TPM devices in your organization will be able to succeed.

Table 4.2 contains a listing of the settings you can implement in your domain using GPOs. Table 4.3 contains the suggested settings you should use.

Table 4.3 Suggested Group Policy Settings in an Active Directory Domain

| GPO Setting | Suggested Setting | Explanation for Setting |
|--|--|---|
| Turn on TPM backup to Active Directory Domain Services | <i>Enabled; Required</i> | You should definitely be backing up the owner authorization values dispersed throughout the systems in your organization to Active Directory to ensure that any required recovery procedures or management tasks requiring owner authorization go smoothly. Make sure the backup is required to ensure that you don't end up with systems for which you don't have the owner authorization values handy due to a change owner task completing without the Active Directory backup completing as well. |
| Configure the list of blocked TPM commands | <i>Enabled; Configure list of commands</i> | We're assuming you have commands you want to block with this one. If not, leave it as Not Configured . You cannot enable this setting and leave the list blank. Once you enable it, you must add at least one command. |
| Ignore the default list of blocked TPM commands | <i>Disabled</i> | You may want to add the default list of blocked commands to your Group Policy list, but you should not use this as a replacement for the default list. No coverage would be provided if a user logged on to a local account. If set to <i>Not Configured</i> , users can set the default list to be ignored using the Local Computer Policy. |
| Ignore the local list of blocked TPM commands | <i>Enabled</i> | Ensure that users cannot configure the local list of blocked commands. |

Deploying TPM-Equipped Devices with Scripting

Once you've configured GPOs to handle your TPM settings, you can move on to using scripts to deploy the devices containing version 1.2 TPMs. We strongly suggest you get everything else in place before moving on to actual device deployment. Update your Active Directory schema if necessary, build your administrative template if necessary, and set the Group Policy settings. This will ensure that once you start taking ownership of TPM chips, the owner authorization values will get backed up to Active Directory. Finally, refer to the section earlier in the chapter where we covered the BIOS settings you need to configure.

Your TPM WMI Primer

It wouldn't be appropriate to simply dive into chunks of script code here without providing a basic explanation of WMI and how you will be able to interact with the TPM using scripts. So, let's start with a WMI primer. We think it will be valuable for all but advanced programmers/scripters.

Microsoft implemented WMI with Windows 2000 in order to serve as a uniform and consistent way to monitor and manage Windows platforms. Not only does it standardize access to components of the systems you need to manage, but it also provides greater access to those components than were previously supported. In other words, administrators were tired of having to manually manage their Windows systems through graphical user interfaces (GUIs) or other kludgy means. There was a strong outcry to be able to script everything, just like those UNIX guys have been doing for the past 30 or so years. In previous versions of Windows, we could use a few command-line interface (CLI) utilities and various scripting languages to hack solutions together, but scripted system management was limited with these tools. Microsoft answered with WMI, which administrators generally rely on VBScript to access.

With WMI, every piece of the system is an object, including files, user accounts, and hardware resources. There is a WMI Class (WMIC) for each type of resource, and each WMIC has methods and properties. You can use scripts to read or modify the properties of an object, and to invoke the methods of the object. WMI remains the consistent interface that exposes those properties and methods to you. The TPM is no different.

Win32_Tpm is the class representing the TPM hardware in your system. Table 4.4 shows some of the more basic and useful methods available with the *Win32_Tpm* class.

Table 4.4 Some Useful Win32_Tpm Methods

| Method | Description |
|-----------------------------------|--|
| <i>AddBlockedCommand</i> | Add a TPM command to the list of commands not permitted to be run by the platform. The command to be blocked is sent with this method as an ordinal value. |
| <i>Clear</i> | This is the equivalent of a pressing a reset button for 5 seconds. It clears all information in the TPM, resetting it back to its factory-default state. |
| <i>Disable</i> | Disables the TPM chip. |
| <i>Enable</i> | Enables the TPM chip. |
| <i>IsActivated</i> | Shows whether the TPM chip is activated. |
| <i>IsCommandBlocked</i> | Shows whether the command queried with this method is listed as blocked. |
| <i>IsEnabled</i> | Shows whether the TPM chip is enabled. |
| <i>IsOwned</i> | Shows whether the TPM chip is owned. |
| <i>RemoveBlockedCommand</i> | Removes a TPM command from the list of commands not permitted to be run by the platform. The command to be removed from the blocked list is sent with this method as an ordinal value. |
| <i>ResetAuthLockOut</i> | Manufacturers implement a lockout period for the TPM to protect against dictionary attacks. This command resets that timeout. |
| <i>TakeOwnership</i> device. | Sets the ownership information for the |
| <i>SetPhysicalPresenceRequest</i> | This method performs functions that require physical presence. The specific command executed depends on the ordinal value sent with the method. This method does not avoid the physical presence requirement for executing the commands. This method sets a device operation as pending, and a reboot is required at which time the platform will display a screen asking the user to accept or reject the operation (see Figure 4.8). |

NOTE

Although you can accomplish a lot with the methods listed in Table 4.4, you should familiarize yourself with the full list of methods and properties in the *Win32_Tpm* class. This will allow you to write more robust scripts. For example, you may want to check whether the device is owned before you go ahead and execute the *TakeOwnership* method. The *Win32_Tpm* class is not overly complex, so you should be able to gain an understanding of the full range of functions fairly quickly. The full *Win32_Tpm* class reference is available at <http://msdn2.microsoft.com/en-us/library/aa376484.aspx>.

Scripting the TPM Deployment

The first task you need to perform when deploying a TPM-equipped platform is to initialize the TPM. This means enabling, activating, and taking ownership of the device. The unfortunate thing when working with TPM-equipped devices is that some tasks require that you be physically present at the machine. This is part of what makes us trust these trusted platforms. So, when you run a script, you still need to be present at the system to provide input when the BIOS requires it with interfaces such as that shown back in Figure 4.8.

In order to initialize the TPM, we suggest that you first use the *SetPhysicalPresenceRequest()* method of the *Win32_Tpm* class. *SetPhysicalPresenceRequest()* allows you to perform a lot of different tasks depending on the numeric value you pass to the method. This method sends a command request to the TPM which is processed on the next reboot. Table 4.5 shows just a handful of these.

Table 4.5 Parameters for the *SetPhysicalPresenceRequest* Method

| Parameter | Description |
|-----------|--|
| 1 | Enable the TPM. |
| 3 | Activate the TPM. |
| 5 | Clear the TPM. |
| 10 | Enable, activate, and allow an owner to be installed on the TPM. |

You probably want to use the *SetPhysicalPresenceRequest()* method as the leadoff in your TPM deployment script. Go ahead and call the method using parameter 10. When the system reboots, you will, as we mentioned, need to be physically present at the device. You should see the screen shown in Figure 4.8 twice before Windows will actually start, and you need to select **MODIFY** for both. Once that is complete, you only need to set an owner for the device. For this, you want to create an SHA-1 hash for the owner authorization value using the *ConvertToOwnerAuth()* method, and then set that value in the device using the *TakeOwnership()* method:

```
'Generate a random number to use as the owner password
Dim num1, num2, pword
num1 = (100000000 * rnd())
num2 = (100000000 * rnd())
pword = (CDBl(num1) * CDBl(num2))

' Create the Win32_Tpm object
Set oTpmService = GetObject("winmgmts:{impersonationLevel=impersonate, " _
                        & "authenticationLevel=pktPrivacy}!\\" _
                        & "." _
                        & "\Root\CIMV2\Security\MicrosoftTpm")
Set oTpm = oTpmService.Get("Win32_Tpm=@")

oTpm.IsActivated isactv      'Test if the TPM is activated
oTpm.IsEnabled isenabled    'Test if the TPM is enabled
oTpm.IsOwnershipAllowed isownable    'Test if ownership of the TPM is
allowed
oTpm.IsOwned isowned        'Test if the TPM is owned

If isowned = False And isenabled = False Then
    oTpm.SetPhysicalPresenceRequest(10)    'Enable, activate, and allow the
installation of an owner
    Wscript.Echo "The TPM has been initialized. The system will now reboot
to complete this operation. When the system reboots a screen will ask if
you want to modify the configuration of the TPM. You will need to allow this
operation twice, and then Windows will load."
    CreateObject("Wscript.Shell").Run "shutdown /r /t 5"    'Reboot the
machine
ElseIf isowned = False And isactv = True And isenabled = True And isownable
= True Then
    oTpm.ConvertToOwnerAuth pword, ownerauth    ' Create a SHA-1 hash of a
password
```

```
oTpm.TakeOwnership(ownerauth)           ' Take ownership
End If
```

Add this command as the logon script for yourself (or a special user account you use to initialize TPM devices), and then start booting and logging on to machines. If these commands complete successfully, the first time you boot you'll get the first set of logical tests. They should indicate that the system needs to be enabled, or activated, or set to allow ownership, and the *SetPhysicalPresenceRequest* call will make it happen. After you answer the BIOS' questions and log on again, the first set of logical tests will fail. The second set will then call the *ConvertToOwnerAuth* and then the *TakeOwnership* methods. After a brief wait, you'll have a TPM that is ready to perform all of the functions that other services and applications that rely upon it need. If your Group Policy is configured as we mentioned earlier, you have all of those TPM owner authorization values stored in Active Directory as well. You may need those if you want to perform further management operations on the TPM.

We won't cover all of the methods of the *Win32_Tpm* class here. You can head to the Microsoft Web site to find references on the class, at <http://msdn2.microsoft.com/en-gb/library/aa376484.aspx>. In this section, we did cover the methods you'll need to deploy TPM systems in your enterprise, however. Just remember that if you need to call a method that utilizes the owner authorization information stored in Active Directory, you need to perform a Lightweight Directory Access Protocol (LDAP) search for *ms-TPM-OwnerInformation* using its LDAP display name, *msTPM-OwnerInformation*. Also, keep in mind that you want to be sure you set up a secure connection to send this information. Do not pass the owner authorization value across your network in plain text.

NOTE

You can rely on the *Win32_Tpm* class alone to perform all your administrative functions, but if you like to have options, there's no need to feel locked in. A script called *manage-bde.wsf* is included with Windows Vista for managing BitLocker Drive Encryption from the command line, and it includes some basic TPM management functions, including the ability to set an owner. So, you could use this script to set an owner by issuing the following command in a command prompt: *cscript manage-bde.wsf -tpm -o owner_password*. The *owner_password* is any string of at least eight characters which the script will automatically hash for you.

TPM Applications

Now that we have seen how the TPM works and we know what sort of capabilities it provides to us, it is time to discuss some of the applications that are using that functionality. Microsoft has made good use of the TPM for some applications in Windows Vista, and some third-party applications that utilize the TPM are already on the market. However, as this is an emerging technology, and because the scope of usefulness of the TPM is so broad, we are likely to see an explosion of applications implementing TPM-based features in the coming years.

Digital Rights Management

The TPM is a part of what Microsoft is calling its Next-Generation Secure Computing Base (NGSCB), or System Integrity, which was originally codenamed Palladium. Many saw the initiative as a way for Microsoft to implement, and to allow others to implement, very strong Digital Rights Management (DRM) protections. This has not changed much in the years since it was first announced, and at this time, there is a great deal of skepticism and an outcry against the NGSCB.

You'll notice that we have not mentioned DRM in this chapter until now, and we've done that for a reason. The TPM, Windows Vista's TPM Services, and the NGSCB overall, provide a great deal of functionality. You can use some of that functionality to implement DRM techniques that are stronger than anything media pirates have come up against thus far. That is sure. However, as you have seen throughout this chapter, the TPM is not about DRM. It is a device centered on cryptography, providing key storage locations, cryptographic functions, and hashing functions. Yes, you can use those cryptography features to implement DRM applications, but you can also use them to implement a great number of security features and applications.

Therefore, we could have spent the entire chapter participating in the flame war that rages on the Internet about the TPM and Microsoft's NGSCB, or we could have a productive discussion about how the TPM works, what it can be used for, and how Windows Vista takes advantage of it. If we had taken the first route, you'd end up with some gross misconceptions about the TPM and everything related to it, and you wouldn't be equipped to implement Windows Vista on TPM-equipped devices.

However, it would be just as misleading if we did not mention DRM at all in the chapter. So, keep in mind that there are DRM applications for the TPM as well. One TPM feature that will be especially useful to those who want to implement DRM is the device authentication feature the TPM provides. You may buy a song via download with a usage right that limits playback of the song to that device

alone. The TPM can seal a key that will be used to encrypt the song, and the song cannot be decrypted and played back from another system.

Microsoft Applications

Currently, Microsoft has implemented a good base of functionality in Windows Vista using the TPM. It has built BitLocker Drive Encryption and its secure startup mechanism around the TPM. It has built a very nice, easy-to-use set of management tools for the TPM in both the TPM MMC and the *Win32_Tpm* WMI class. However, there is a whole set of TPM-based functionality, called Code Integrity, that is available only in the 64-bit version of Windows Vista.

When we discussed the TCG trusted platform, we mentioned that the TPM could be used to extend trust to higher-level components that run on the OS. Code Integrity takes advantage of this by implementing the following protections:

- Verifies the integrity of all code that loads into a protected process.
- Winload verifies the integrity of all drivers that are critical to the boot process, including the HAL and the Windows kernel.
- Verifies the integrity of all kernel-mode drivers.
- Verifies the integrity of all user-mode binaries that implement cryptographic functions.
- Verifies the integrity of all user-mode binaries that load into a protected process used for the playback of high-definition media.
- Verifies the integrity of a specific set of user-mode binaries using page hashes in *nt5ph.cat* and *ntpe.cat*.
- Verifies kernel code.

These functions are generally carried out just as we discussed in the beginning of the chapter: Code integrity measurements are taken, and the TPM can be used to attest to these measurements. This protects the system from running any of the areas of code mentioned earlier if intentional or unintentional corruption occurs. If a rootkit successfully loads into the kernel on your Windows Vista 64-bit system, it will be detected, and it will be prevented from executing. Obviously, this does not necessarily protect your system from being compromised by a rootkit, but it does ensure that the compromise is brought to light when integrity measurements uncover it.

Third-Party Applications

Some third-party applications that rely on the TPM have already begun to emerge. Probably the best example of this is Wave Embassy Suites. This software package was originally developed to take advantage of version 1.1 TPM chips, but it now supports version 1.2 chips as well. This is a very popular application that OEMs are deploying with devices they sell that include biometric hardware. The TPM will be utilized by applications such as this which enable strong biometric authentication measures by securing the biometric data that the application relies upon. Just as the TPM assists BitLocker Drive Encryption by sealing the VMK, the TPM can seal the user's biometric data that the application is using.

Another important application for TPM-enabled devices that will emerge will be remote access solutions. Given the strong device authentication possible with the TPM, you can be sure that remote access will appeal to a segment of the software market that turns toward implementing TPM support. In this way, you can think of the user needing an RSA SecureID or smart card to authenticate to the remote access solution, and the device he or she is connecting from will also need to present its credentials using the TPM. This can help network administrators eliminate remote security breaches due solely to compromised user authentication tokens such as passwords, RSA SecureIDs, or smart cards. Now the attacker will also need to have the user's laptop as well.

This is really only the tip of the iceberg as far as TPM-based applications go, however. Some implementations we are likely to see are as follows:

- Users are provided with the ability to wrap the cryptographic keys they use for secure connections over the Internet, such as logging into their bank accounts via a Web browser and using secure e-mail applications.
- Web servers can use the TPM's sealing functionality to provide assurance to connecting nodes that they are trustworthy, including assurance that the server-side software and settings are the same as when the node connected previously, when the trust relationship was established.
- DRM will likely emerge as a popular way to use the TPM. Media applications such as iTunes can use the TPM's sealing functions to lock down access to media if tampering takes place.

Understanding the Security Implications of the TPM

One of the main implications of the TPM is that it will require massive hardware expenditures in order to take advantage of it. In most cases, when a new version of Windows is released, every enterprise has some systems that need to be upgraded in order to run the new version, but many of their existing systems already support it. So, the rollout does not require sweeping hardware upgrades across the board. However, the TPM is a new piece of hardware that is still included in only certain product offerings from OEMs, with market penetration of these devices only really getting started within the past 12 months, and it is required in order to take advantage of the TPM services in Windows Vista. This means enterprises that want to take advantage of the TPM will need to budget for widespread replacements of desktop and mobile systems, which can be expensive.

Aside from the budgetary implications, the TPM and TPM support in Windows Vista solve many security problems, and at the same time there is a lot of nonsense about the TPM being a magic bullet. These issues are worth discussing so that we, as information security professionals, can make informed decisions about what countermeasures are worth implementing, and so that we can accurately assess the security posture of our systems and our enterprise as a whole.

Encryption as a Countermeasure

Let's start by first mentioning cryptography in a general sense. After all, the TPM device and TPM services in Windows Vista are obviously focused on encryption. The TPM is mainly designed to create, store, and protect encryption keys, password hashes, and digital certificates.

The first thing we should say here is that no single countermeasure will fully protect digital assets from attack. In fact, there is probably not a combination of countermeasures that can ensure 100 percent protection from attack. Cryptography is no exception to that rule. Consider this as you read vendor product claims, white papers, and other industry articles that tout TPM as the end of your search for a secure enterprise.

Several sources are currently touting media sanitization as a feature of the TPM. The argument goes that you can clear the TPM in a matter of seconds, and your data is gone. No more spending money on and waiting hours for National Security Agency (NSA) media sanitization procedures to work. Just clear the TPM and your data is gone forever. There are two important points here:

1. When you clear the TPM your data is *not* gone. Only the key that was used to wrap the key that was used to encrypt that data is gone. The data, in encrypted form, still exists on the drive.
2. Encryption can be defeated by brute force attacks.

This is not meant to argue that encryption is insecure. The fact of the matter is that with current computing power, it would take a state-of-the-art desktop machine more than 100 trillion years to brute force your data if it is encrypted using the 128-bit Advanced Encryption Standard (AES)! Obviously, if it takes trillions of years to crack into your data, it's safe. We raise this issue only because a lot of talk is circulating about using the clearing of the TPM as a media sanitization method, when *technically speaking*, the data is not destroyed and it can be attacked using brute force. Now those brute force attacks may very well remain infeasible for the next 2,000 years. However, there is a chance that leaps in the power of computers over the next two decades or a flaw discovered in the AES algorithm could make brute forcing a 128-bit AES encryption seem as easy as cracking the Data Encryption Standard (DES) is today.

Tools & Traps...

Media Sanitization

The most important thing to consider when using encryption is how long the data you are protecting needs to remain confidential. If you have information on your hard drive that you expect to remain sensitive for the next six months (maybe it is a proposal you are working on, and you must turn it in within that six-month time frame), you can expect the AES algorithm with a 128-bit key to do the job for you. On the other hand, if you are storing a full portfolio of personal information which you need to keep confidential for the next 50 years, you might want to consider how quickly a computer can chug through those keys 35 years down the road. What is the shelf life of your data?

Even the good folks at the National Institute of Standards and Technology (NIST) seem to be riding the fence on this media sanitization issue. In the recently released Special Publication 800-88, "Guidelines for Media Sanitization," found at http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf, NIST added and then removed the following text:

Continued

“Encryption is not a generally accepted means of sanitization. The increasing power of computers decreases the time needed to crack cipher text and therefore the inability to recover the encrypted data [cannot] be assured.”

You’ll also find a lot of good debate on the topic throughout the Internet. We don’t usually disagree with what comes out of NIST, and we won’t start here. Just remember that this method of sanitization does not destroy the data or the media. Encryption is a method of ensuring that it takes a long time to get to sensitive data, and for most data this method of sanitization will probably buy us enough time. However, in some applications, this will not be an adequate sanitization method. The important point is that you know how sensitive your data is and how long it must remain confidential before it becomes useless, and that you apply the proper methods given those facts.

You should also note that NIST did not go so far as to add cryptographic destruction anywhere in its list of sanitization methods.

Now that we’ve addressed that caveat, let’s look at the power at our fingertips. Where the TPM and its capability to give us a more robust encryption platform help the most is simply in the fact that the data we are protecting is increasingly moving outside the walls of the enterprise. As we mentioned before, employees are going ever more mobile, with laptops replacing desktops, and cell phones, PDAs, and even MP3 players providing more and more storage and remote connectivity features. Those firewalls, proxies, DMZs, and other layered perimeter protection rings we designed and built in the past are doing a great job at keeping the bad stuff out. What we have increasingly less ability to do is keep the assets we protect in.

This is where encryption and the trusted platform come to the rescue. Encrypting data, especially when we utilize full-disk encryption technologies such as BitLocker Drive Encryption, allows us to create an environment where employees are carrying our security perimeter with them wherever they go. That encryption creates a boundary wherever the device and the data are between the data and the outside world. There are, of course, already software-only solutions for full-disk encryption, but the TPM provides better protection of encryption keys because key recovery techniques that may have worked fairly well against keys held in software are not likely to work against keys protected by the TPM.

Our protection is strengthened even more by the fact that many normal attacks which involve subverting the system software will not work. As soon as part of the system has been modified, the chain of trust will cease to extend to that part of the system. The platform will not load, and the data will not be able to be recovered by simply creating a backdoor in some poorly coded software. The only piece of software we have to rely upon is the tiny bit of code known as the CRTM. Because this

code is small and relatively simple, it is easier to ensure that there are no vulnerabilities in it. Buffer overflows and backdoors tend to get lost in a program of tens of thousands of lines of code.

Either brute force cracking methods or attacks on the hardware itself will be required. As long as the data was encrypted using large keys and a secure algorithm, brute force attacks will prove to be a fruitless endeavor, and hardware attacks require a lot more skill and resources than running the canned attack code available on the Internet which is used against software.

Notes from the Underground...

Mandating Full-Disk Encryption

On June 23, 2006, the Office of Management and Budget (OMB) issued Memoranda M-06-16, "Protection of Sensitive Agency Information." This was a direct response to many of the data loss problems that afflicted the federal government during the first half of 2006. The memoranda requires a blend of technical, management, and operational controls defined in the NIST Special Publication (SP) 800-53, "Recommended Security Controls for Federal Information Systems," to be implemented, but one specific requirement leading off the memoranda requires that all sensitive data on mobile devices be encrypted.

In rapid response to the memoranda, the U.S. Air Force posted a request for a full-disk encryption solution on the Federal Business Opportunity Web site. The U.S. Department of Agriculture (USDA) also posted a request for quotes (RFQ) on www.fbo.gov in relation to mobile device encryption. In an article for *Government Computer News* found at www.gcn.com/online/vol1_no1/42640-1.html?topic=mobile-wireless, Mary Mosquera wrote that the USDA's requirements for the encryption solution include the following:

- It must be Federal Information Processing Standards (FIPS) 140-2 compliant.
- It must integrate with a Microsoft Active Directory infrastructure.
- It must be invisible to users.
- It must be scalable.
- It must provide automated deployment tools.
- It must provide adequate recovery processes.

Continued

As we have seen throughout this chapter, devices containing version 1.2 TPM chips and Windows Vista TPM services fulfill all of these requirements. They go beyond the requirements by securing encryption keys with TPM hardware instead of relying on a software-only solution. Whether Windows Vista becomes the solution of choice for most federal agencies remains to be seen.

This is the leading edge of a widespread change in the way that both the public and the private sectors protect their digital assets on mobile devices. As the federal government leads this process of carefully identifying all PII, controlling which mobile devices that the PII. It is just unfortunate that the compromise of the PII of millions of people has been the impetus required to effect this change.

Here are some references for more information on this issue:

- You can find OMB Memoranda M-06-16 at www.whitehouse.gov/omb/memoranda/fy2006/m06-16.pdf.
- You can find information about the U.S. Air Force procurement at www.fbo.gov/spg/USAF/AFMC/ESC/FA8771-07-R-0001/Attachments.html.
- You can find NIST SP 800-53 Revision 1 at <http://csrc.nist.gov/publications/nistpubs/800-53-Rev1/800-53-rev1-final-clean-sz.pdf>.

Can I Really Trust These People?

We must rely on trustworthy BIOSes, TPM drivers, the TBS, and TSS implementations from myriad sources in order to be sure that our trusted platforms are actually trustworthy. More important, as we saw in discussing the trusted platform architecture, the platform builds by evaluating the trustworthiness of each component one at a time, and then relying on that component to evaluate the next component, and so on. In this way, trust in one component is derived from the trust we had in each preceding component. Therefore, if one component in the chain is suspect, all components that derived their trust from that piece are also suspect.

How much can we really trust these platforms, then? We can be sure that there will be numerous problems with code security in the coming years. BIOSes, drivers, OSes, and third-party applications will all be susceptible to attack. What we rely upon is the capability of the trusted platform to provide reliable metrics about itself—in fact, the TCG has defined *trusted platform* as “a computing platform that can be trusted to report its properties,” in its glossary of terms at www.trustedcomputinggroup.org/groups/glossary. So, the idea is not that the trusted computing platform is impervious to attack, nor that it can detect when it has been compromised. As we

discussed at the beginning of this chapter, the trusted platform can enter insecure or unstable states. However, it must provide reliable measurements of itself. Other parties then challenge the system to produce these measurements when they need to decide whether they trust the platform, and in this way, communication with an untrusted system can be avoided.

The TPM Only Enables Technical Security Controls

Any good information security program relies upon technical, operational, and management security controls. The TPM and Windows Vista TPM services provide us with an incredibly powerful and flexible set of technical controls. However, they do not help us to implement operational or management controls, and therefore they can never be touted as the end game in information security. So, be wary of any claims of that nature.

What the TPM and TPM services in Windows Vista do allow us to do is to implement a range of controls based on encryption and device authentication to greatly improve enterprise security. However, as with many of the technical controls we currently have, holes still exist. For the most part, people are either using or implementing those technical controls, and we have management and operational controls to which those people are supposed to adhere. Whether they do adhere to them is the issue.

For example, we read in the sidebar about best practices for TPM management that the TPM owner password should never be stored to local media. However, when we initialized the TPM and set the TPM owner password using the automatically created password, we were required to save the password to local media in an XML file with a .tpm extension (refer to the section on initializing the TPM, and to Figure 4.2). From Microsoft's perspective, this required save is understandable. There is simply no way that anyone will remember a 48-digit password, so in order to ensure that the owner authorization information is available when required, Microsoft requires that automatically created authorization information be saved.

This is exactly where a good operational control is necessary. That file should be stored in a secured location away from the device on which it was created as soon as the initialization process completes. This will eliminate the chance that the owner authorization information is stolen along with the asset to which the owner authorization information pertains. However, those TPM files will invariably be saved to USB keys which will then be carried around in the laptop bag or backpack that contains the laptop on which they were created. So, basically, we have a tool that allows us to provide a high level of security for a platform if we use it correctly, but

in all likelihood, a simple mistake such as this will invalidate any security that the TPM can provide.

This is not a weakness of the TPM or of Windows Vista's TPM services. This should be considered a weakness of an immature security program that does not provide a broad range of controls, and instead foolishly counts on a single magic bullet to provide protection. With an already robust security program, the TPM could be just what is needed to help the organization implement an even stronger security posture.

Are You Owned?

I Wonder How Much My Social Security Number Costs

If you believe that technical security controls are superior to management and/or operational controls, please revisit the "Are You Owned?" sidebar that kicked off this chapter. Note that ChoicePoint *sold* PII on more than 160,000 people to identity thieves just a few years ago. ChoicePoint was not hacked. There was no missing laptop. Identity thieves were able to breach very weak management and operational controls. They portrayed themselves as private companies that required the information for various credit and background checks, and ChoicePoint sold them the information despite the fact that policies and procedures for performing basic checks would have revealed the fraud. The following quote is from an MSNBC.com article on the data breach, found at www.msnbc.msn.com/id/11030692:

"The FTC's complaint against ChoicePoint paints a picture of a firm that was selling data to all comers, even after obvious signs of trouble. Law enforcement agencies began to warn ChoicePoint of fraudulent activity back in 2001, the complaint alleges. ChoicePoint continued to sell data to companies with expired business licenses, with canceled telephones and after employees signaled them out as suspicious. The firm even continued to supply credit reports to the crime ring after the fake accounts it had set up were suspended by ChoicePoint for non-payment, the complaint says."

Existing Attacks

The main problem people may have with Windows Vista TPM services is incorrect usage of it. As we discussed, many people see clearing the TPM as a valid method of wiping your drive if the drive is encrypted using the TPM. The flip side to that is

that it takes only one small mistake to eliminate all of someone's data. So be careful with the TPM MMC. Do not run the Clear TPM wizard if you have data that you need that was encrypted with the TPM and be careful with those scripts. When you issue a *SetPhysicalPresenceRequest(5)* call on a machine, you had better mean it, or you had better be able to get on the machine and prevent the configuration modification from completing after reboot.

By the same token, this may become a popular form of denial of service (DoS) attack for the bad guys. If they cannot get to your data, it may simply be enough to prevent you from getting to it. This means organizations are going to make sure they enable the backup of recovery keys so that they can enact emergency recovery procedures. This makes protecting the Active Directory infrastructure more important than ever, but we already have ways to protect what is inside the perimeter. We need to have a tool that could help us extend the security perimeter outside the walls of the office, and that is where we can leverage the TPM to great effect.

What we will see is how effective and easy hardware attacks can become. We know this is a weakness, now that we are relying on hardware to provide some security functions. These sorts of attacks generally require more skill and are harder to implement than software attacks, though, and attackers will always take the path of least resistance.

Summary

The TPM is the cornerstone of the TCG trusted platform, which is a computing platform that can provide reliable integrity metrics on itself. The TPM itself does not prevent virus attacks, theft of equipment, theft of data, or hacking attempts.

However, it does allow software developers to outfit security professionals, administrators, and even users with a wide range of tools that can protect their systems.

Windows Vista includes some of these tools, including BitLocker Drive Encryption, a secure startup mechanism within BitLocker, and Code Integrity features. The TPM and Windows TPM services also support strong device authentication, which gives network administrators a reliable means for controlling connectivity throughout their networks.

It is interesting to see that although Microsoft has certainly spent a lot of development time on implementing TPM services and applications in Windows Vista, it also has spent a very large amount of time and effort on penetration testing Windows Vista's TPM implementation. We included a link to the Doug MacIver presentation earlier in the chapter. Microsoft used BitLocker penetration testing as a way to provide feedback to the developers, and apparently this has had an important impact on the choices of which PCRs are used to seal the VMK.

This is an emerging technology, in terms of both the hardware and the software that takes advantage of it. At this point, it seems as though Microsoft has built a very robust and secure platform around the TPM, but given the wide-ranging possibilities for TPM applications, we have seen only the tip of the iceberg. Although Microsoft has even taken a whole new approach to system architecture in controlling what processes may operate in kernel mode, and implementing Code Integrity to provide integrity monitoring for all of that code, Code Integrity could very well be extended to provide this integrity checking for other parts of the system. As Tom Petty once sang, "the future was wide open."

However, it's good to bring at least a small degree of skepticism anytime you examine something. As we pointed out throughout this chapter, as old attack surfaces vanish, attackers will find new ones. Hardware attacks such as those that are currently possible against smart cards will emerge as a popular target. Although the TPM is just emerging, some lessons are being incorporated into chip design, such as under- and over-volting protection mechanisms. Despite any weaknesses, the TPM and Windows Vista's TPM services provide security professionals with a very useful tool to secure their data.

Solutions Fast Track

Understanding the TPM

- ☑ The Trusted Computing Group is an industry standards organization that is developing specifications for the trusted platform architecture. The TPM is at the core of the trusted platform.
- ☑ Trusted platforms are based on two trusted components: the TPM and CRTM, which are called the Trusted Building Blocks. Trust in the rest of the platform is derived from these two basic components. The trust boundary gradually extends to include other components, such as the OS and applications.

Configuring and Managing the TPM on a Stand-Alone System

- ☑ Use the TPM MMC console to configure the TPM on your stand-alone system. This MMC provides all the functionality you should need in a familiar interface that is easy to use.
- ☑ Always back up your TPM owner authorization information to an external storage device, and make sure you do not keep this device with the system for which it contains the owner authorization information.

Configuring and Managing the TPM in an Enterprise Environment

- ☑ Make sure you are requiring that the TPM owner authorization information is backed up to Active Directory, if at all possible. This backup functionality requires (1.) that all your domain controllers are running Windows Server 2003 SP1 or later and (2.) that you have upgraded your Active Directory schema using the adprep utility that comes with the Windows Server 2007 and Windows Vista DVDs.
- ☑ Utilize the Group Policy settings covered earlier in this chapter to lock down users' ability to tamper with the TPM command block lists, and to configure your central block list. If you need to have the Group Policy

settings available with Windows Server 2007 on your Windows Server 2003 domain controllers, you can use the code included in this chapter and on the CD that comes with this book to modify your administrative templates.

- ☑ Use scripting to take advantage of the *Win32_Tpm* WMI class to ease your TPM device deployments. You can refer to Microsoft's reference documentation on this class at <http://msdn2.microsoft.com/en-gb/library/aa376484.aspx> in order to familiarize yourself with the class.

TPM Applications

- ☑ Microsoft has built several key TPM-related components into Windows Vista. The TBS has been implemented to serve as an agent that mediates access to the TPM. The TCG has outlined an architecture whereby a trusted platform relies on the BIOS and the OS boot manager to implement a trusted boot process in order to maintain system integrity through to the OS. BitLocker Drive Encryption implements this trusted boot process. See the coverage of BitLocker Drive Encryption provided in Chapter 5.
- ☑ A small number of applications rely on the TPM, and there should be large growth in these types of applications once Windows Vista is officially released and begins to gain a foothold in desktop deployments.
- ☑ To the dismay of music and movie lovers everywhere, the TPM will enable content providers to implement more robust DRM techniques.

Understanding the Security Implications of the TPM

- ☑ The TPM and Windows Vista TPM services are powerful tools for securing the enterprise. They can provide very strong device authentication, powerful protection of encryption keys, and assurance that code running on the system is trustworthy. However, the TPM and services that depend on it cannot ensure security. In order to provide security, we as security professionals must implement strong technical, management, and operational controls. The TPM can help us to implement strong technical controls, but it does not address the other control areas.
- ☑ As small devices include ever-increasing storage capacity, information security professionals have two problems to solve as users become more

mobile. First, we must understand the data we protect so that we know where any sensitive data is, and we must provide policies and training on how the data is to be stored and handled. Second, we must implement a mobile security perimeter to protect that data when it leaves the walls of the enterprise, and the way to do this is to use cryptography.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Can I protect sensitive data by storing it in the TPM in my computer?

A: No, the TPM is used only to create and store the private portions of keys, and certain platform metrics. Those keys may then be used to encrypt the contents of a disk or other data storage locations. So, you would use the TPM to protect your data, but not by directly storing the data in the TPM.

Q: If I want to use Microsoft’s new BitLocker Drive Encryption, do I need to have a TPM?

A: No, you can take advantage of BitLocker Drive Encryption by storing the encryption key on a USB storage device. However, using the TPM to store the key is preferred, and it is strongly recommended that you utilize the TPM if one is present in your system. For more information on BitLocker Drive Encryption, see Chapter 5.

Q: Does the TPM mean that Windows Vista is hack-proof?

A: No, nothing can make a computer hack-proof. Even if a system is unplugged and powered off it is susceptible to physical attacks. However, the TPM and Windows Vista TPM services have provided coverage against a lot of the most popular current attack vectors, and using them together will provide better security than an otherwise identical system that does not take advantage of them can provide. In the meantime, attackers are going to be looking for new attack surfaces through which they can gain access to the system and the data stored on it. Implementing layers of defense across the spectrum of technical, management,

and operational security controls is a necessary supplement to a security program that leverages the powerful countermeasures that Windows Vista's TPM services provide.

Q: Are there any differences between the TPM features included in the 32-bit version and 64-bit version of Windows Vista?

A: Yes, there are. Only the 64-bit version of Windows Vista includes Code Integrity features, which include:

- Verification of the integrity of all code that loads into a protected process
- Winload verification of the integrity of all drivers which are critical to the boot process, including the HAL and the Windows kernel
- Verification of the integrity of all kernel-mode drivers
- Verification of the integrity of all user-mode binaries that implement cryptographic functions
- Verification of the integrity of all user-mode binaries that load into a protected process used for the playback of high-definition media
- Verification of the integrity of a specific set of user-mode binaries using page hashes in nt5ph.cat and ntpe.cat
- Verification of kernel code

