CHAPTER

20 Operating System Security Models

In this chapter, we will discuss concepts related to operating system security models, namely

- The security reference monitor and how it manages the security of its related elements
- Access control—the heart of information security
- International standards for operating system security, which provide organizations with a level of assurance and integrity

Quite simply, an operating system security model is the foundation of the operating system's security functionality. All security functionality is architected, specified, and detailed in advance—before a single line of code is written. Everything built on top of the security model must be mapped back to it, and any action that violates the security model should be denied and logged.

Operating System Models

The *operating system security model* (also known as the *trusted computing base,* or TCB) is simply the set of rules, or protocols, for security functionality. Security commences at the network protocol level and maps all the way up to the operations of the operating system.

An effective security model protects the entire host and all of the software and hardware that operate off it. Previous systems used an older, monolithic design, which proved to be less than effective. Current operating systems are optimized for security, using a compartmentalized approach.

The trend in operating systems has been toward a microkernel architecture. In contrast to the monolithic kernel, microkernels are platform independent. Although they lack the performance of monolithic systems, they are catching up in terms of speed and optimization. A microkernel approach is built around a small kernel with a common hardware level. The key advantage of a microkernel is that the kernel is small and easy to port to other systems.

From a security perspective, by using a microkernel or compartmentalized approach, you contain any capacity to inflict damage to the system. This is akin to the watertight sections of a submarine; if one section is flooded, it can be sealed and the submarine can still operate. But this works only when the security model is well defined and tested. If not, then this approach would be more analogous to the Titanic; the ship was considered unsinkable because it was compartmentalized, but in the end, it was sunk by an iceberg because the compartments were flooded when they were overloaded.

The Underlying Protocols Are Insecure

Extending the submarine analogy, the security protocol has a direct connection to the communication protocol. Today, the protocol is TCP/IP—the language of the Internet and, clearly, the most popular and utilized protocol. If the operating system is an island, then TCP/IP is the sea. Given that fact, any operating system used today must make up for TCP/IP's shortcomings.

Even the best operating system security model can't operate in a vacuum or as an island, however. If the underlying protocols are insecure, then the operating system is at risk. What's frightening about this insecurity is that while the language of the Internet is TCP/IP, effective security functionality was not added to TCP/IP until version 6 in the late 1990s. Given that the vast majority of the Internet is still running an insecure version of TCP/IP, version 4 (version 5 was never put into production), the entire Internet and corporate computing infrastructure is built on and running on an insecure infrastructure and foundation.

We've known about TCP/IP's lack of security for a long time. The protocol's main problems are as follows:

- **Vulnerable to spoofing** Spoofing is the term for establishing a connection with a forged sender address. Normally this involves exploiting trust relations between the source address and the destination address. The ability to spoof the source IP address assists those carrying out DoS attacks by making it difficult for victims to block the DoS traffic, and the predictability of the initial sequence number (ISN), which is a unique number that is supposed to guarantee the authenticity of the sender, contributes more to spoofing attacks by allowing an attacker to impersonate legitimate systems and take over a connection (as in a man in the middle attack).
- **Vulnerable to session hijacking** An attacker can take control of a connection by intercepting the session key and using it to insert his own traffic into an established TCP/IP communication session, usually in combination with a DoS attack against the legitimate sender so that traffic cannot get through, as in a man in the middle attack.
- **Predictable sequence guessing** The sequence number used in TCP connections is a 32-bit number, so the odds of guessing the correct ISN would seem to be exceedingly low. If the ISN for a connection is assigned in a predictable way, however, it becomes relatively easy to guess. The truth is that the ISN problem is not a protocol problem but rather an implementation problem. The protocol actually specifies pseudorandom sequence numbers, but many implementations have ignored this recommendation.

- **No authentication or encryption** The lack of authentication and encryption with TCP/IP is a major weakness.
- **Vulnerable to SYN flooding** SYN flooding takes advantage of the three-way handshake in establishing a connection. When Host B receives a SYN request from A, it must keep track of the partially opened connection in a *listen queue,* enabling successful connections even with long network delays. The problem is that many implementations can keep track of only a limited number of connections. A malicious host can exploit the small size of the listen queue by sending multiple SYN requests to a host but never replying to the SYN and ACK the other host sends back. By doing so, the malicious host quickly fills up the other host's listen queue, and that host stops accepting new connections until a partially opened connection in the queue is completed or times out.

If you want a more detailed look at the myriad security issues with TCP/IP version 4, read Steve Bellovin's seminal paper "Security Problems in the TCP/IP Protocol Suite," the classic resource for this issue.

The security benefits of TCP/IP version 6 include

- IPSec security
- Authentication and encryption
- Resilience against spoofing
- Data integrity safeguards
- Confidentiality and privacy

An effective security model recognizes and is built around the fact that because security is such an important design goal for the operating system, every resource that the operating system interfaces with (memory, files, hardware, device drivers, and so on) must interact from a security perspective. By giving each of these objects an access control list (ACL), the operating system can detail what that object can and can't do by limiting its privileges.

Access Control Lists

Much of the security functionality afforded by an operating system is via the ACL. Access control comes in many forms, but in whatever form it is implemented, it is the foundation of any security functionality.

Access control enables you to protect a server or parts of the server (directories, files, file types, and so on). When the server receives a request, it determines access by consulting a hierarchy of rules in the ACL.

An access control list is defined as a table that tells a computer operating system which access rights each user has to a particular system object, such as a file directory or an individual file. Each object has a security attribute that identifies its access control list. The list has an entry for each system user with access privileges. The most common privileges include the ability to read a file (or all the files in a directory), to write to the file or files, and to execute the file (if it is an executable file or program). Each operating system implements the ACL differently.

In Windows, an ACL is associated with each system object. Each ACL has one or more *access control entries (ACEs),* each consisting of the name of a user or a group of users. The user can also be a role name, such as *programmer* or *tester.* For each of these users, groups, or roles, the access privileges are stated in a string of bits called an *access mask.* Generally, the system administrator or the object owner creates the access control list for an object.

Each ACE identifies a security principal and specifies a set of access rights that are allowed, denied, or audited for that security principal. An object's security descriptor can contain two ACLs:

- A *discretionary* access control list (DACL) that identifies the users and groups who are allowed or denied access
- A *system* access control list (SACL) that controls how access is audited

Unix systems also have access control based on user permissions and roles defined by groups. System objects have permissions defined within them, which can be controlled on the basis of read, write, and execute permissions for each user or group defined on the system.

MAC vs. DAC

Access control lists can be further refined into both required and optional settings. This refinement is carried out more precisely with *discretionary access control* and is implemented by discretionary access control lists (DACLs). The difference between discretionary access control and its counterpart, mandatory access control, is that DAC provides an entity or object with access privileges it can pass to other entities. Depending on the context in which they are used, these controls are also called rule-based access control (RBAC) and identity-based access control (IBAC).

Mandatory access control requires that access control policy decisions be beyond the control of the individual owners of an object. MAC is generally used in systems that require a very high level of security. With MAC, only the administrator and *not* the owner of the resource may make decisions that bear on or derive from the security policy. Only a security administrator may change a resource's category, and no one may grant a right of access that is explicitly forbidden in the access control policy.

MAC is always prohibitive (i.e., all that is not expressly permitted is forbidden) and not permissive. Only within that context do discretionary controls operate, prohibiting still more access with the same exclusionary principle.

All of the major operating systems (Solaris, Windows, NetWare, and so on) use DAC. MAC is implemented in more secure, trusted operating systems such as TrustedBSD and Trusted Solaris.

Table 20-1 details the difference in functionality between discretionary and mandatory access control.

Control Type	Functionality
Discretionary	—Individual users may determine the access controls. —Works well in commercial and academic sector. —Not suited for the military. —Effective for private web sites, etc.
Mandatory	—Allows the system administrator to set up policies and accounts that will allow each user to have full access to the files and resources needed, but no access to other information and resources not immediately necessary to perform assigned tasks. —Site-wide security policy is enforced by the system in addition to the discretionary access controls. —Better suited to environments with rigid information. —Effective access restrictions. —Access permission cannot be passed from one user to another. —Requires labeling: sensitivity and integrity labels.

Table 20-1 The Difference in Functionality Between Discretionary and Mandatory Access Control

Classic Security Models

Anyone who has studied for the CISSP exam or studied postgraduate computer security knows that three of the most famous security models are Bell-LaPadula, Biba, and Clark-Wilson. These three models are often mentioned in computing textbooks, and they form the foundation of most current operating system models. But practically speaking, most of them are little used in the real world, functioning only as security references.

Those designing operating system security models have the liberty of picking and choosing from the best of what the famous models have, without being encumbered by their myriad details.

Bell-LaPadula

While the Bell-LaPadula model was revolutionary when it was published in 1976, descriptions of its functionality today are almost anticlimactic. The Bell-LaPadula model was one of the first attempts to formalize an information security model. The Bell-LaPadula model was designed to prevent users and processes from reading above their security level. This is used within a data classification system—so a given classification cannot read data associated with a higher classification—as it focuses on sensitivity of data according to classification levels.

In addition, this model prevents objects and processes with any given classification from writing data associated with a lower classification. This aspect of the model caused a lot of consternation in the security space. Most operating systems assumed that the need to write below one's classification level is a necessary function. But the military influence on which Bell-LaPadula was created mandated that this be taken into consideration.

In fact, Bell-LaPadula's connection to the military is so tight that much of the TCSEC (aka the Orange Book, described further below) was designed around Bell-LaPadula.

Biba

Biba is often known as a reversed version of Bell-LaPadula, as it focuses on integrity labels, rather than sensitivity and data classification. (Bell-LaPadula was designed to keep secrets, not to protect data integrity.)

Biba covers integrity levels, which are analogous to sensitivity levels in Bell-LaPadula, and the integrity levels cover inappropriate modification of data. Biba attempts to preserve the first goal of integrity, namely to prevent unauthorized users from modifying data.

Clark-Wilson

Clark-Wilson attempts to define a security model based on accepted business practices for transaction processing. Much more real-world-oriented than the other models described, it articulates the concept of *well-formed transactions* that

- Perform steps in order
- Perform exactly the steps listed
- Authenticate the individuals who perform the steps

TCSEC

In the early 1970s, the United States Department of Defense published a series of documents to classify the security of operating systems, known officially as the *Trusted Systems Security Evaluation Criteria* and unofficially (but more commonly) as the rainbow series (www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html). The TCSEC was heavily influenced by Bell-LaPadula and classified systems at levels *A* through *D*.

TCSEC was developed to meet three objectives:

- To give users a yardstick for assessing how much they can trust computer systems for the secure processing of classified or other sensitive information
- To guide manufacturers in what to build into their new, widely available commercial products to satisfy trust requirements for sensitive applications
- To provide a basis for specifying security requirements for software and hardware acquisitions

Table 20-2 provides a brief overview of the different classification levels.

Although TCSEC offered a lot of functionality, it was, by and large, not suitable for the era of client/server computing. The client/server computing world was embryonic when the TCSEC was created, although its objectives were admirable. Neither Microsoft nor Intel was really on the scene, and no one thought that one day a computer would be on every desktop. C2 is a dated, military-based specification that does not work well in the corporate computing environment. Basically, it doesn't address critical developments in high-level computer security, and it is cumbersome to implement in networked systems.

For those who wanted to go beyond Orange Book functionality to their networked systems, they had to apply the requirements of the *Trusted Network Interpretation of the TCSEC (TNI),* also known as the *Red Book.*

TCSEC Rating	Usage
D—Minimal Protection	—Any system that does not comply with any other category or has failed to receive a higher classification —No security requirements —Was used as a catch-all category for such operating systems as MS-DOS and Windows 95/98/ME
C1—Discretionary Protection	—DACL/ACL—User/Group/World protection —Usually for users who are all on the same security level —Protected operating system and system operations mode —Periodic integrity checking of TCB —Tested security mechanisms with no obvious bypasses —Documentation for user security —Documentation for systems administration security —Documentation for security testing —TCB design documentation
C2—Controlled Access Protection	Everything in C1 plus: —Object protection can be on a single-user basis, for example, through an ACL or trustee database —Authorization for access may be assigned only by authorized users —Object reuse protection —Mandatory identification and authorization procedures for users, such as username/password —Full auditing of security events —Protected system mode of operation —Added protection for authorization and audit data —Documentation as C1 plus information on examining audit information —One of the most common certifications, including VMS, IBM OS/400, Windows NT 3.51, Novell NetWare 4.11, Oracle 7, and DG AOS/VS II
B1—Labeled Security Protection	Everything in C2 plus: —Mandatory security and access labeling of all objects, for example, files, processes, devices —Label integrity checking (for example, maintenance of sensitivity labels when data is exported) —Auditing of labeled objects —Mandatory access control for all operations —Enhanced auditing —Enhanced protection of operating systems —Improved documentation —Operating systems: HP-UX BLS, Cray Research Trusted Unicos 8.0, Digital SEVMS, Harris CS/SX, and SGI Trusted IRIX *(continued)*

Table 20-2 Classifications of Operating Systems Security

TCSEC Rating	**Usage**
B2—Structured Protection	Everything in B1 plus: —Notification of security level changes affecting interactive users —Hierarchical device labels —Mandatory access over all objects and devices —Trusted path communications between user and system —Tracking down of covert storage channels —Tighter system operations mode into multilevel independent units —Covert channel analysis —Improved security testing —Formal models of TCB —Version, update, and patch analysis and auditing —Example systems: Honeywell Multics and Trusted XENIX
B3—Security Domains	Everything in B2 plus: —ACL additionally based on groups and identifiers —Trusted path access and authentication —Automatic security analysis —TCB models more formal —Auditing of security auditing events —Trusted recovery after system down and relevant documentation —Zero design flaws in TCB and minimum implementation flaws —Only B3-certified OS is Getronics/Wang Federal XTS-300
A1—Verified Design	A1 is the highest level of certification and demands a formal security verification method to ensure that security controls protect classified and other sensitive information. At this level, even the National Security Agency cannot break in. A1 requires everything in B3 plus: —Formal methods and proof of integrity of TCB —Only A1-certified systems: Gemini Trusted Network Processor and Honeywell SCOMP

Table 20-2 Classifications of Operating Systems Security *(continued)*

Finally, the coupling of assurance and functionality is really what brought down the TCSEC. Most corporate environments do not have enough staff to support the assurance levels that TCSEC required. Also, the lack of consideration of networks and connectivity also played a huge role, as client/server computing is what brought information technology into the mainstream. Nonetheless, a positive outcome of the TCSEC standards was that they formed the basis and key conceptual building blocks for many of today's standards.

Labels

TCSEC makes heavy use of the concept of *labels*. Labels are simply security-related information that has been associated with objects such as files, processes, or devices. The ability to associate security labels with system objects is also under security control.

Sensitivity labels, used to define the level of data classification, are composed of a sensitivity level and possibly some number of sensitivity categories. The number of sensitivity levels available is dependent on the specific operating system.

In a commercial environment, the label attribute could be used to classify, for example, levels of a management hierarchy. Each file or program has one hierarchical sensitivity level. A user may be allowed to use several different levels, but only one level may be used at any given time.

While the sensitivity labels identify whether a user is cleared to view certain information, *integrity* labels identify whether data is reliable enough for a specific user to see. An integrity label is composed of an integrity grade and some number of integrity divisions. The number of hierarchical grades to classify the reliability of information is dependent on the operating system.

While TCSEC requires the use of labels, other regulations and standards such as the Common Criteria (Common Criteria for IT Security Evaluation, ISO Standard 15408) also require security labels.

There are many other models around, including the Chinese wall (seeks to prevent information flow that can cause a conflict of interest), Take-Grant (a model that helps in determining the protection rights, for example, read or write, in a computer system), and more. But in practice, none of these models has found favor in contemporary operating systems (Linux, Unix, Windows)—they are overly restrictive and reflect the fact that they were designed before the era of client/server computing.

Current operating system architects are able to use these references as models, pick and choose the best they have to offer, and design their systems accordingly.

Reference Monitor

In this section, we discuss the *reference monitor* concept and how it fits into today's security environment.

The Computer Security Technology Planning Study Panel called together by the United States Air Force developed the reference monitor concept in 1972. They were brought together to combat growing security problems in a shared computer environment. In 1972, they were unable to come up with a fail-safe solution; however, they were responsible for reshaping the direction of information security today.

The Reference Monitor Concept

The National Institute of Standards and Technologies describes the reference monitor concept as an object that maintains the access control policy. It does not actually change the access control information; it only provides information about the policy.

The security reference monitor is a separable module that enforces access control decisions and security processes for the operating system. All security operations are routed through the reference monitor, which decides if the specific operation should be permitted or denied.

Perhaps the main benefit of a reference model is that it can provide an abstract model of the required properties that the security system and its access control capabilities must enforce.

The main elements of an effective reference monitor are that it is

- **Always on** Security must be implemented consistently and at all times for the entire system and for every file and object.
- **Not subject to preemption** Nothing should be able to preempt the reference monitor. If this were not the case, then it would be possible for an entity to bypass the mechanism and violate the policy that must be enforced.

- **Tamperproof** It must be impossible for an attacker to attack the access mediation mechanism such that the required access checks are not performed and authorizations not enforced.
- **Lightweight** It must be small enough to be subject to analysis and tests, proving its effectiveness.

Although few reference models have been used in their native state, as Cynthia Irvine of the Naval Postgraduate School writes in "The Reference Monitor Concept as a Unifying Principle in Computer Security Education," for over 25 years, the reference monitor concept has proved itself to be a useful tool for computer security practitioners. It can also be used as a conceptual tool in computer security education.

Windows Security Reference Monitor

The Windows Security Reference Monitor (SRM) is responsible for validating Windows process access permissions against the security descriptor for a given object. The Object Manager then, in turn, uses the services of the SRM while validating the process's request to access any object.

Windows is clearly not a bulletproof operating system, as is evident from the number of security advisories alone. In fact, it is full of security holes. But the fact that it is the most popular operating system in use in corporate settings and that Microsoft has been, for the most part, open with its security functionality, makes it a good case study for a real-world example of how an operating system security model should operate.

Trustworthy Computing

For many years, people would never use *Microsoft* and *security* in the same sentence. But all of that started to change in early 2002 with Microsoft's *Trustworthy Computing* initiative. On January 15, 2002, Bill Gates sent a memo to all Microsoft employees stating that security was the highest priority for all the work Microsoft was doing.

The four goals of the Trustworthy Computing initiative are

- **Security** As a customer, you can expect to withstand attack. In addition, you can expect the data is protected to prevent availability problems and corruption.
- **Privacy** You have the ability to control information about yourself and maintain privacy of data sent across the network.
- **Reliability** When you need your system or data, they are available.
- **Business integrity** The vendor of a product acts in a timely and responsible manner, releasing security updates when a vulnerability is found.

To track and assure its progress in complying with the Trustworthy Computing initiative, Microsoft created a framework to explain its objectives: that its products be secure by design, secure by default, and secure in deployment, and that it provide communications (SD3+C).

Secure by design simply means that all vulnerabilities are resolved prior to shipping the product. Secure by design requires three steps.

1. *Build a secure architecture.* This is imperative. Software needs to be designed with security in mind first and then features.

2. *Add security features.* Feature sets need to be added to deal with new security vulnerabilities.
3. *Reduce the number of vulnerabilities in new and existing code.* The internal process at Microsoft was revamped to make developers more conscious of security issues while designing and developing software.

Secure in deployment means ongoing protection, detection, defense, recovery, and maintenance through good tools and guidance.

Communications is the key to the whole project. How quickly can Microsoft get the word out that a vulnerability exists and help you to understand how to operate your system with enhanced security?

International Standards for Operating System Security

Although Microsoft's Trustworthy Computing initiative has been heralded as a giant step forward for computer security, much of the momentum started years earlier. And one of the prime forces has been the Common Criteria.

The need for a common information security standard is obvious. Security means many things to different people and organizations. But this subjective level of security cannot be objectively valuable. Therefore, common criteria were needed to evaluate the security of an information technology product.

Common Criteria

The need for common agreement is clear. When you buy a DVD, put gas in your car, or make a purchase from an online retailer, all of these activities function because they operate in accordance with a common set of standards and guidelines.

And that is precisely what the Common Criteria are meant to be, a global security standard ensuring that there is a common mechanism for evaluating the security of technology products and systems. By providing a common set of requirements for comparing the security functions of software and hardware products, the Common Criteria enable users to have an objective yardstick by which to evaluate a product's security.

Common Criteria certification is slowly but increasingly being used as a touchstone for many Requests for Proposals, primarily in the government sector. By offering a consistent, rigorous, and independently verifiable set of evaluation requirements for hardware and software, Common Criteria certification is intended to be the Good Housekeeping seal of approval for the information security sector.

But what is especially historic about the Common Criteria is that this is the first time governments around the world have united in support of an information security evaluation program.

Common Criteria Origins

In the United States, the Common Criteria have their roots in the Trusted Computer System Evaluation Criteria (TCSEC), also known as the Orange Book. But by the early 1990s, it was clear that TCSEC was not viable for the new world of client/server computing. Its main problem was that it was not accommodating to new computing paradigms.

And with that, TCSEC as it was known is dead. The very last C2 and B1 Orange Book evaluations performed by the NSA under the Orange Book itself were completed and publicly announced at the NISSC conference in October 2000. The C2 and B1 classes (see "TCSEC" earlier in the chapter and Table 20-2) have been converted to protection profiles under the Common Criteria, however, and C2 and B1 evaluations are still being performed by commercial laboratories under the Common Criteria. According to the TPEP web site, NSA is still willing to perform Orange Book evaluations at B2 and above, but most vendors prefer to evaluate against newer standards cast as Common Criteria protection profiles.

Another subtle point is that the Orange Book and the Common Criteria are not exactly the same types of documents. Whereas the Orange Book is a set of requirements that reflect the practice and policies of a specific community (the U.S. Department of Defense and later the national security community), the Common Criteria are policy-independent and can be used by many organizations (including those in the DoD and the NSA) to articulate their security requirements.

In Europe, the Information Technology Security Evaluation Criteria (ITSEC), already in development in the early 1990s, were published in 1991 by the European Commission. This was a joint effort with representatives from France, Germany, the Netherlands, and the United Kingdom contributing.

Simultaneously, the Canadian government created the Canadian Trusted Computer Product Evaluation Criteria as an amalgamation of the ITSEC and TCSEC approaches. In the United States, the draft of the Federal Criteria for Information Technology Security was published in 1993, in an attempt to combine the various methods for evaluation criteria.

With so many different approaches going on at once, there was consensus to create a common approach. At that point, the International Organization for Standardization (ISO) began to develop a new set of standard evaluation criteria for general use that could be used internationally. The goal was to unite the various international and diverse standards into new criteria for the evaluation of information technology products. This effort ultimately led to the development of the Common Criteria, now an international standard in ISO 15408:1999. (The official name of the standard is the *International Common Criteria for Information Technology Security.*)

Common Criteria Sections

Common Criteria is a set of three distinct but related parts. These are the three parts of the Common Criteria:

- Part 1 is the introduction to the Common Criteria. It defines the general concepts and principles of information technology security evaluation and presents a general model of evaluation. Part 1 also presents the constructs for expressing information technology security objectives, selecting and defining information technology security requirements, and writing high-level specifications for products and systems. In addition, the usefulness of each part of the Common Criteria is described in terms of each of the target audiences.
- Part 2 details the specific security functional requirements and details a criterion for expressing the security functional requirements for Targets of Evaluation (TOE).
- Part 3 details the security assurance requirements and defines a set of assurance components as a standard way of expressing the assurance requirements for TOE. Part 3 lists the set of assurance components, families, and classes and defines evaluation criteria for protection profiles (PPs). A protection profile is a set of

security requirements for a category of TOE and security targets (STs). Security targets are the set of security requirements and specifications to be used as the basis for evaluating an identified TOE. Part 3 also presents evaluation assurance levels that define the predefined Common Criteria scale for rating assurance for TOE, namely the evaluation assurance levels (EALs).

Protection Profiles and Security Targets

Protection profiles (PPs) and security targets (STs) are two building blocks of the Common Criteria.

A *protection profile* defines a standard set of security requirements for a specific type of product (for example, operating systems, databases, or firewalls). These profiles form the basis for the Common Criteria evaluation. By listing required security features for product families, the Common Criteria allow products to state conformity to a relevant protection profile. During Common Criteria evaluation, the product is tested against a specific PP, providing reliable verification of the product's security capabilities.

The overall purpose of Common Criteria product certification is to provide end users with a significant level of trust. Before a product can be submitted for certification, the vendor must first specify a security target. The *security target* description includes an overview of the product, potential security threats, detailed information on the implementation of all security features included in the product, and any claims of conformity against a PP at a specified EAL.

The vendor must submit the ST to an accredited testing laboratory for evaluation. The laboratory then tests the product to verify the described security features and evaluate the product against the claimed PP. The end result of a successful evaluation includes official certification of the product against a specific protection profile at a specified evaluation assurance level.

Problems with the Common Criteria

Although there are benefits to the Common Criteria, there are also problems with this approach. The point of this section is not to detail those problems, but in a nutshell, to give you a brief summary of the issues:

- **Administrative overhead** The overhead involved with gaining certification takes a huge amount of time and resources.
- **Expense** Gaining certification is extremely expensive.
- **Labor-intensive certification** The certification process takes months.
- **Need for skilled and experienced analysts** Availability of information security professionals with the required experience is still lacking.
- **Room for various interpretations** The Common Criteria leave room for various interpretations of what the standard is attempting to achieve.
- **Paucity of Common Criteria testing laboratories** There are only seven laboratories in the United States.
- **Length of time to become a Common Criteria testing laboratory** Even for those organizations that are interested in becoming certified, the process in and of itself takes quite a while.

Summary

In this chapter, we explored different security models for operating systems, including the classics: Bell-LaPadula, Biba, Clark-Wilson, and TCSEC. These classic models ultimately led to today's operating system security standards.

We also saw how the security reference monitor is a critical aspect to the underlying operating system's security functionality. Because all security functionality is architected, specified, and detailed in the operating system, it is the foundation to all security above it. Understanding how this functionality works, and how it is tied specifically to the operating system used within your organization, is crucial to ensuring that information security is maximized.

Finally, we discussed the Trustworthy Computing initiative, international standards for operating system security, and the Common Criteria—its origins, sections, protection profiles, security targets, and shortcomings.

References

Bach, Maurice. *The Design of the UNIX Operating System*. Prentice Hall, 1986.

Comer, Douglas. *Operating System Design: The XINU Approach*. Addison-Wesley, 1983.

Crowley, Charles. *Operating Systems: A Design-Oriented Approach*. Irwin Professional Publishing, 1996.

Deitel, Harvey, Paul Deitel, and David Choffnes. *Operating Systems*. Prentice Hall, 2003.

McKusick, Marshall, and George Neville-Neil. *The Design and Implementation of the FreeBSD Operating System*. Addison-Wesley, 2004.

Russinovich, Mark, David Solomon, and Alex Ionescu. *Windows Internals*. Microsoft Press, 2009.

Silberschatz, Avi, Peter Galvin, and Greg Gagne. *Operating System Concepts*. Wiley, 2012.

Stallings, William. *Operating Systems: Internals and Design Principles*. Prentice Hall, 2011.

Tanenbaum, Andrew. *Modern Operating Systems*. Prentice Hall, 2007.

Tanenbaum, Andrew, and Albert Woodhull. *Operating Systems Design and Implementation*. Prentice Hall, 2006.