**Objects and Subjects**

Controlling access to any resource in a secure system involves two entities. The *subject* is the user or process that makes a request to access a resource. Access can mean reading from or writing to a resource. The *object* is the resource a user or process wants to access.

This also serves as an example of transitive trust. Transitive trust is the concept that if A trusts B and B trusts C, then A inherits trust of C through the transitive property—which works like it would in a mathematical equation: if a = b, and b = c, then a = c.

**Closed and Open Systems**

Systems are designed and built according to one of two differing philosophies: **A closed system is designed to work well with a narrow range of other systems, generally all from the same manufacturer. The standards for closed systems are often proprietary and not normally disclosed**. **Open systems, on the other hand, are designed using agreed-upon industry standards. Open systems are much easier to integrate with systems from different manufacturers that support the same standards.**

Closed systems are harder to integrate with unlike systems, but they can be more secure. A closed system often comprises proprietary hardware and software that does not incorporate industry standards. This lack of integration ease means that attacks on many generic system components either will not work or must be customized to be successful**. In many cases, attacking a closed system is harder than launching an attack on an open system.**

Open systems are generally far easier to integrate with other open systems. It is easy, for example, to create a LAN with a Microsoft Windows Server machine, a Linux machine, and a Macintosh machine. Although all three computers use different operating systems and could represent up to three different hardware architectures, each supports industry standards and makes it easy for networked (or other) communications to occur. This ease comes at a price, however. Because standard communications components are incorporated into each of these three open systems, there are far more predictable entry points and methods for launching attacks. In general, their openness makes them more vulnerable to attack, and their widespread availability makes it possible for attackers to find (and even to practice on) plenty of potential targets. Also, open systems are more popular than closed systems and attract more attention. An attacker who develops basic attacking skills will find more targets on open systems than on closed ones. This larger “market” of potential targets usually means that there is more emphasis on targeting open systems. Inarguably, there’s a greater body of shared experience and knowledge on how to attack open systems than there is for closed systems.

**Confinement**

Software designers use process confinement to restrict the actions of a program. Simply put, process **confinement allows a process to read from and w t rite to only certain memory locations and resources**. This is also **known as sandboxing.**

**Bounds**

Each process that runs on a system is assigned an authority level. The authority level tells the operating system what the process can do. In simple systems, there may be only two authority levels: user and kernel. **The authority level tells the operating system how to set the bounds for a process. The bounds of a process consist of limits set on the memory addresses and resources it can access. The bounds state the area within which a process is confined or contained**.

**Isolation**

When a process is confined through enforcing access bounds, that process runs in isolation. Process isolation ensures that any behavior will affect only the memory and resources associated with the isolated process. Isolation is used to protect the operating environment, the kernel of the OS, and other independent applications. Isolation is an essential component of a stable operating system. Isolation is what prevents an application from accessing the memory or resources of another application, whether for good or ill. The operating system may provide intermediary services, such as cut-and-paste and resource sharing (such as the keyboard, network interface, and storage device access). These three concepts (confinement, bounds, and isolation) make designing secure programs and operating systems more difficult, but they also make it possible to implement more secure systems.

**Controls**

To ensure the security of a system, you need to allow subjects to access only authorized objects. A control uses access rules to limit the access of a subject to an object.

There are both **mandatory and discretionary access controls, often called MAC and DAC, respectively**

With mandatory controls, static attributes of the subject and the object are considered to determine the permissibility of an access**. Each subject possesses attributes that define its clearance, or authority, to access resources. Each object possesses attributes that define its classification. Different types of security methods** classify resources in different ways. For example, subject A is granted access to object B if the security system can find a rule that allows a subject with subject A’s clearance to access an object with object B’s classification. This is called rule-based access control (RBAC). The predefined rules state which subjects can access which objects.

**Discretionary controls differ from mandatory controls in that the subject has some ability to define the objects to access**. Within limits, discretionary access controls allow the subject to define a list of objects to access as needed. This access control list serves as a dynamic access rule set that the subject can modify. The constraints imposed on the modifications often relate to the **subject’s identity. Based on the identity**, the subject may be allowed to add or modify the rules that define access to objects. Both mandatory and discretionary access controls limit the access to objects by subjects. The primary goal of controls is to ensure the confidentiality and integrity of data by disallowing unauthorized access by authorized or unauthorized subjects.

**Trust and Assurance**

A trusted system is one in which all protection mechanisms work together to process sensitive data for many types of users while maintaining a stable and secure computing environment. **Assurance is simply defined as the degree of confidence in satisfaction of security needs. Assurance must be continually maintained, updated, and reverified.**

Several different methods are used to describe the necessary security attributes for an object. **A security token is a separate object that is associated with a resource and describes its security attributes. This token can communicate security information about an object prior to requesting access to the actual object**. In other implementations, various lists are used to store security information about multiple objects. A **capabilities list maintains a row of security attributes for each controlled object. Although not as flexible as the token approach, capabilities lists generally offer quicker lookups when a subject requests access to an** object. A third common type of attribute storage is called **a security label, which is generally a permanent part of the object t l o which it’s attached. Once a security label is set, it** usually cannot be altered. This permanence provides another safeguard against tampering that neither tokens nor capabilities lists provide.

**Trusted Computing Base**

trusted computing base (TCB) as a combination of hardware, software, and controls that work together to form a trusted base to enforce your security policy. The TCB is a subset of a complete information system. It should be as small as possible so that a detailed analysis can reasonably ensure that the system meets design specifications and requirements. The TCB is the only portion of that system that can be trusted to adhere to and enforce the security policy. It is not necessary that every component of a system be trusted. But any time you consider a system from a security standpoint, your evaluation should include all trusted components that define that system’s TCB.

**In general, TCB components in a system are responsible for controlling access to the system. The TCB must provide methods to access resources both inside and outside the TCB itself**. TCB components commonly restrict the activities of components outside the TCB. It is the responsibility of TCB components to ensure that a system behaves properly in all cases and that it adheres to the security policy under all circumstances

**Security Perimeter**

The security perimeter of your system is an imaginary boundary that separates the TCB from the rest of the system (This boundary ensures that no insecure communications or interactions occur between the TCB and the remaining elements of the computer system. **For the TCB to communicate with the rest of the system, it must create secure channels, also called trusted paths**. A trusted path is a channel established with strict standards to allow necessary communication to occur without exposing the TCB to security vulnerabilities. A trusted path also protects system users (sometimes known as subjects) from compromise as a result o s f a TCB interchange. As you learn more about formal security guidelines and evaluation criteria later in this chapter, you’ll also learn that trusted paths are required in systems that seek to deliver high levels of security to their users. According to the TCSEC guidelines**, trusted paths are required for high trust level systems such as those at level B2 or higher of TCSEC.**

**Reference Monitors and Kernels**

When the time comes to implement a secure system, it’s essential to develop some part of the TCB to enforce access controls on system assets and resources (sometimes known as objects). **The part of the TCB that validates access to every resource prior to granting access requests is called the reference monitor.**

The reference monitor stands between every subject and object, verifying that a requesting subject’s credentials meet the object’s access requirements before any requests are allowed to proceed. If such access requirements aren’t met, access requests are turned down. Effectively, the reference monitor is the access control enforcer for the TCB. Thus, authorized and secured actions and activities are allowed to occur, whereas unauthorized and insecure activities and actions are denied and blocked from occurring. **The reference monitor enforces access control or authorization based the desired security model, whether discretionary, mandatory, role-based, or some other form of access control. The reference monitor may be a conceptual part of the TCB; it doesn’t need to be an actual, stand-alone, or independent working system** component

**The collection of components in the TCB that work together to implement reference monitor functions is called the security kernel** . The reference monitor is a concept or theory that is put into practice via the implementation of a security kernel in software and hardware. **The purpose of the security kernel is to launch appropriate components to enforce reference monitor functionality and resist all known attacks**. The security kernel uses a trusted path to communicate with subjects. It also mediates all resource access requests, granting only those requests that match the appropriate access rules in use for a system. The reference monitor requires descriptive information about each resource that it protects. Such information normally includes its classification and designation. When a subject requests access to an object, the reference monitor consults the

object’s descriptive information to discern whether access should be granted or denied.

**State Machine Model**

**The state machine model describes a system that is always secure no matter what state it is in.** It’s based on the computer science definition of a finite state machine (FSM). An FSM combines an external input with an internal machine state to model all kinds of complex systems, including parsers, decoders, and interpreters. Given an input and a state, an FSM transitions to another state and may create an output. Mathematically, the next state is a function of the current state and the input next state; that is, the next state = F (input, current state). Likewise, the output is also a function of the input and the current state output; that is, the output = F (input, current state).

Many security models are based on the secure state concept. A transition occurs when accepting input or producing output. A transition always results in a new state (also called a state transition). If each possible state transition results in another secure state, the system can be called a secure state machine.

**Information Flow Model**

**The information flow model focuses on the flow of information. Information flow models are based on a state machine model**. The Bell-LaPadula and Biba models, which we will discuss in detail later in this chapter, are both information flow models**. Bell-LaPadula is concerned with preventing information flow from a high security level to a low security level**. **Biba is concerned with preventing information flow from a low security level to a high security level**. Information flow models don’t necessarily deal with only the direction of information flow; they can also address the type of flow.

**Information flow models are designed to prevent unauthorized, insecure, or restricted information flow, often between different levels of security (these are often referred to as multilevel models).** Information flow can be between **subjects and objects at the same classification level as well as between subjects and objects at different classification levels**. An information flow model allows all authorized information flow, whether within the same classification level or between classification levels. **It prevents all unauthorized information flow, whether within the same classification level or between classification levels**. Another interesting perspective on the information flow model is **that it is used to establish a relationship between two versions or states of the same object when those two versions or states exist at different points** in time. Thus, information flow dictates the transformation of an object from one state at one point in time to another state at another point in time. **The information flow model also addresses covert channels by**

**specifically excluding all non-defined flow pathways**.

**Noninterference Model**

**The noninterference model is loosely based on the information flow model.** However, instead of being concerned about the flow of information, **the noninterference model is concerned with how the actions of a subject at a higher security level affect the system state or the actions of a subject at a lower security level**. Basically, the actions of subject A (high) should not affect the actions of subject B (low) or even be noticed by subject B. **The real concern is to prevent the actions of subject A at a high level of security classification from affecting the system state at a lower level. If this occurs, subject B may be placed into an insecure state or be able to deduce or infer information about a higher level of classification. This is a type of information leakage and implicitly creates a covert channel.** Thus, the noninterference model can be imposed to provide a form of protection against damage caused by malicious programs such as Trojan horses.

**Take-Grant Model**

The Take-Grant model employs a directed graph to dictate how rights can be passed from one subject to another or from a subject to an object. Simply put, a subject with the grant right can grant another subject or another object any other right they possess. Likewise, a subject with the take right can take a right from another subject. In addition to these two primary rules, the Take-Grant model may adopt a create rule and a remove rule to generate or delete rights. The key to this model is that using these rules allows you to figure out when rights in the system can change and where leakage (that is, unintentional distribution of permissions) can occur.

Take rule Allows a subject to take rights over an object

Grant rule Allows a subject to grant rights to an object

Create rule Allows a subject to create new rights

Remove rule Allows a subject to remove rights it has.

**Access Control Matrix**

**An access control matrix is a table of subjects and objects that indicates the actions or functions that each subject can perform on each object.** **Each column of the matrix is an access control list (ACL).** **Each row of the matrix is a capabilities list . An ACL is tied to the object; it lists valid actions each subject can perform. A capability list is tied to the subject; it lists valid actions that can be taken on each object**. From an administration perspective, using only capability lists for access control is a management nightmare. A capability list method of access control can be accomplished by storing on each subject a list of rights the subject has for every object. This effectively gives each user a key ring of accesses and rights to objects within the security domain. To remove access to a particular object, every user (subject) that has access to it must be individually manipulated. Thus, managing access on each user account is much more difficult than managing access on each object (in other words, via ACLs).

Implementing an access control matrix model usually involves the following:

**Constructing an environment that can create and manage lists of subjects and objects**

**■ Crafting a function that can return the type associated with whatever object is supplied to that function as input (this is important because an object’s type determines what kind of operations may be applied to it)**

**Bell-LaPadula Model**

**The US Department of Defense (DoD) developed the Bell-LaPadula model in the 1970s to address concerns about protecting classified information. The DoD manages multiple levels of classified resources, and the Bell-LaPadula multilevel model was derived from the DoD’s multilevel security policies.** The classifications the DoD uses are numerous; however, discussions of classifications within the CISSP CBK are usually limited **to unclassified, sensitive but unclassified, confidential, secret, and top secret.** The multilevel security policy states that a subject with any level of clearance can access resources at or below its clearance level. However, within the higher clearance levels, access is granted only on a need-to know basis. In other words, access to a specific object is granted to the classified levels only if a specific work task requires such access. For example, any person with a secret security clearance can access secret, confidential, sensitive but unclassified, and unclassified documents but not top-secret documents. Also, to access a document within the secret level, the person seeking access must also have a need to know for that document.

By design, the Bell-LaPadula model prevents the leaking or transfer of classified information to less secure clearance levels. This is accomplished by blocking lower-classified subjects from accessing higher-classified objects. With these restrictions, **the Bell-LaPadula model is focused on maintaining the confidentiality of** objects. Thus, the complexities involved in ensuring the confidentiality of documents are addressed in the Bell-LaPadula model. However, **Bell-LaPadula does not address the aspects of integrity or availability for** objects. **Bell-LaPadula is also the first mathematical model of a multilevel security policy.**

**Subjects under lattice-based access controls are assigned positions in a lattice. Lattice-based access controls also fi t into the general category of information flow models and deal primarily with confidentiality (that’s the reason for the connection to Bell-LaPadula**).

**The Simple Security Property states that a subject may not read information at a higher sensitivity level (no read up).**

**■ The \* (star) Security Property states that a subject may not write information to an object at a lower sensitivity level (no write down). This is also known as the Confinement Property.**

**■ The Discretionary Security Property states that the system uses an access matrix to**

**enforce discretionary access control.**

These first two properties define the states into which the system can transition. No other transitions are allowed. All states accessible through these two rules are secure states. Thus, Bell-LaPadula–modeled systems offer state machine model security.

**An exception in the Bell-LaPadula model states that a “trusted subject” is not constrained by the \* Security Property. A trusted subject is defined as “a subject that is guaranteed not to consummate a security-breaching information transfer even if it is possible.” This means that a trusted subject is allowed to violate the \* Security Property and perform a write down, which is necessary when performing valid object declassification or reclassification**.

The Bell-LaPadula properties are in place to protect data confidentiality. A subject cannot read an object that is classified at a higher level than the subject is cleared for. Because objects at one level have data that is more sensitive or secret than data in objects at a lower level, a subject (who is not a trusted subject) cannot write data from one level to an object at a lower level. That action would be similar to pasting a top-secret memo into an unclassified document file. **The third property enforces a subject’s need to know in order to access an object.**

**Biba Model**

For many nonmilitary organizations, integrity is more important than confidentiality. Out of this need, several integrity-focused security models were developed, such as those developed by Biba and by Clark-Wilson. The Biba model was designed after the Bell-LaPadula model. Where the Bell-LaPadula model addresses confidentiality, the Biba model addresses integrity. The Biba model is also built on a state machine concept, is based on information flow, and is a multilevel model. In fact, Biba appears to be pretty similar to the Bell-LaPadula model, except inverted. Both use states and transitions. Both have basic properties. The biggest difference is their primary focus: **Biba primarily protects data integrity**. Here are the basic properties or axioms of the Biba model state machine:

The Simple Integrity Property states that a subject cannot read an object at a lower integrity level (no read-down).

■ The \* (star) Integrity Property states that a subject cannot modify an object at a higher integrity level (no write-up).

**Take note that simple is always about reading, and star is always about writing.**

Biba was designed to address three integrity issues:

■ Prevent modification of objects by unauthorized subjects.

■ Prevent unauthorized modification of objects by authorized subjects.

■ Protect internal and external object consistency.

Critiques of the Biba model reveal a few drawbacks:

■ It addresses only integrity, not confidentiality or availability.

■ It focuses on protecting objects from external threats; it assumes that internal threats are handled programmatically.

■ It does not address access control management, and it doesn’t provide a way to assign or change an object’s or subject’s classification level.

■ **It does not prevent covert channels.**

Because the Biba model focuses on data integrity, it is a more common choice for commercial security models than the Bell-LaPadula model. Most commercial

**Clark-Wilson Model**

The Clark-Wilson model uses a multifaceted approach to enforcing data integrity. Instead of defining a formal state machine, the Clark-Wilson model defines each data item and allows modifications through only a small set of programs.

**The Clark-Wilson model does not require the use of a lattice structure; rather, it uses a three-part relationship of subject/program/object (or subject/transaction/object) known as a triple or an access control triple. Subjects do not have direct access to objects. Objects can be accessed only through programs. Through the use of two principles—well-formed transactions and separation of duties—the Clark-Wilson model provides an effective means to protect integrity**.

**Well-formed transactions take the form of programs**. A subject is able to access objects only by using a program, interface, or access portal. **Each program has specific limitations on what it can and cannot do to an object (such as a database or other resource). This effectively limits the subject’s capabilities. This is known as a constrained interface. If the programs are properly designed, then the triple relationship provides a means to protect the integrity of the object**.

Clark-Wilson defines the following items and procedures:

■ A constrained data item (CDI) is any data item whose integrity is protected by the security model.

An unconstrained data item (UDI) is any data item that is not controlled by the security model. Any data that is to be input and hasn’t been validated, or any output, would be considered an unconstrained data item.

■ An integrity verification procedure (IVP) is a procedure that scans data items and confirms their integrity.

■ **Transformation procedures (TPs) are the only procedures that are allowed to modify a CDI.** The limited access to CDIs through TPs forms the backbone of the Clark-Wilson integrity model.

**The Clark-Wilson model uses security labels to grant access to objects, but only through transformation procedures and a restricted interface model.** A restricted interface model uses classification-based restrictions to offer only subject-specific authorized information and functions. One subject at one classification level will see one set of data and have access to one set of functions, whereas another subject at a different classification level will see a different set of data and have access to a different set of functions. The different functions made available to different levels or classes of users may be implemented by either showing all functions to all users but disabling those that are not authorized for a specific user or by showing only those functions granted to a specific user. Through these mechanisms, **the Clark-Wilson model ensures that data is protected from unauthorized changes from any user. In effect, the Clark-Wilson model enforces separation of duties. The Clark-Wilson design makes it a good model for commercial applications.**

**Brewer and Nash Model (aka Chinese Wall)**

**This model was created to permit access controls to change dynamically based on a user’s previous activity (making it a kind of state machine model as well). This model applies to a single** integrated database; it seeks to create security domains that are sensitive to the notion of conflict of interest (for example, someone who works at Company C who has

access to proprietary data for Company A should not also be allowed access to similar data for Company B if those two companies compete with each other). This model is known as the Chinese Wall because it creates a class of data that defines which security domains are potentially in conflict and prevents any subject with access to one domain that belongs to a specific conflict class from accessing any other domain that belongs to the same conflict class. Metaphorically, this puts a wall around all other information in any conflict class. Thus, this model also uses the principle of data isolation within each conflict class to keep users out of potential conflict-of-interest situations (for example, management of company datasets). Because company relationships change all the time, dynamic updates to members of and definitions for conflict classes are important. Another way of looking at or thinking of the Brewer and Nash model is of an administrator having full control access to a wide range of data in a system based on their assigned job responsibilities and work tasks. However, at the moment an action is taken against any data item, the administrator’s access to any conflicting data items is temporarily blocked. Only data items that relate to the initial data item can be accessed during the operation. Once the task is completed, the administrator’s access returns to full control.

**Goguen-Meseguer Model**

**The Goguen-Meseguer model is an integrity model, although not as well known as Biba and the others**. In fact**, this model is said to be the foundation of noninterference conceptual theories. Often when someone refers to a noninterference model, they are actually referring to the Goguen-Meseguer model**. The Goguen-Meseguer model is **based on predetermining the set or domain—a list of objects that a subject can access. This model is based on automation theory and domain separation. This means subjects are allowed only to perform predetermined actions against predetermined objects.** When similar users are grouped into their own domain (that is, collective), the members of one subject domain cannot interfere with the members of another subject domain. Thus, subjects are unable to interfere with each other’s activities.

**Sutherland Model**

**The Sutherland model is an integrity model. It focuses on preventing interference in support of integrity. It is formally based on the state machine model and the information flow model. However, it does not directly indicate specific mechanisms for protection of integrity. Instead, the model is based on the idea of defining a set of system states, initial states, and state transitions.** Through the use of only these predetermined secure states, integrity is maintained and interference is prohibited**. A common example of the Sutherland model is its use to prevent a covert channel from being used to influe**nce the outcome of a process or activity.

**Graham-Denning Model**

The Graham-Denning model **is focused on the secure creation and deletion of both subjects and objects**. Graham-Denning is a collection of eight primary protection rules or actions that define the boundaries of certain secure actions:

■ Securely create an object.

■ Securely create a subject.

■ Securely delete an object.

■ Securely delete a subject.

■ Securely provide the read access right.

■ Securely provide the grant access right.

■ Securely provide the delete access right.

■ Securely provide the transfer access right.

Usually the specific abilities or permissions of a subject over a set of objects is defined in an access matrix

Systems Security Evaluation Models

When formal evaluations are undertaken, systems are usually subjected to a two-step process:

1. The system is tested and a technical evaluation is performed to make sure that the system’s security capabilities meet criteria laid out for its intended use.

2. The system is subjected to a formal comparison of its design and security criteria and its actual capabilities and performance, and individuals responsible for the security and veracity of such systems must decide whether to adopt them, reject them, or make some changes to their criteria and try again.

Often trusted third parties are hired to perform such evaluations; the most important result from such testing is their “seal of approval” that the system meets all essential criteria. Regardless of whether the evaluations are conducted inside an organization or out of house, the adopting organization must decide to accept or reject the proposed systems. An organization’s management must take formal responsibility if and when a system is adopted and be willing to accept any risks associated with its deployment and use.

The three main product evaluation models or classification criteria models addressed here **are TCSEC, ITSEC, and Common Criteria.**

You should be aware that TCSEC was repealed and replaced by the Common Criteria

**Rainbow Series**

The first such set of standards resulted in the creation of the Trusted Computer System Evaluation Criteria (TCSEC) in the 1980s, as the US Department of Defense (DoD) worked to develop and impose security standards for the systems **it purchased and used. In turn, this led to a whole series of such publications through the mid-1990s. Since these publications were routinely identified by the color of their covers, they are known collectively as the rainbow series.**

Significant standards in this group include a **European model called the Information Technology Security Evaluation Criteria (ITSEC),** which was developed in 1990 and used through 1998. **Eventually TCSEC and ITSEC were replaced with the so-called Common Criteria, adopted by the United States, Canada, France, Germany**, and the United Kingdom in 1998 but **more formally known as the “Arrangement on the Recognition of Common Criteria Certificates in the Field of IT Security.”**

When governments or other security-conscious agencies evaluate information systems, they make use of various standard evaluation criteria. In 1985, **the National Computer Security Center (NCSC) developed the TCSEC, usually called the Orange Book because of the color of this publication’s covers**. **The TCSEC established guidelines to be used when evaluating a stand-alone computer from the security perspective**. These guidelines **address basic security functionality and allow evaluators to measure and rate a system’s functionality and trustworthiness.** **In the TSCEC, in fact, functionality and security assurance are combined and not separated as they are in security criteria** developed later. TCSEC guidelines were designed to be used when evaluating vendor products or by vendors to ensure that they build all **necessary functionality and security assurance into new products.**

**TCSEC Classes and Required Functionality**

**TCSEC combines the functionality and assurance rating of the confidentiality protection** offered by a system into four major categories. **These categories are then subdivided into additional subcategories identified with numbers, such as C1 and C2**. Furthermore, TCSEC’s categories are assigned through the evaluation of a target system. Applicable systems are stand-alone systems that are not networked. TCSEC defines the following major categories

**Category A Verified protection. The highest level of security.**

**Category B Mandatory protection.**

**Category C Discretionary protection.**

**Category D Minimal protection. Reserved for systems that have been evaluated but do**

**not meet requirements to belong to any other category.**

**Level Label Requirements**

D Minimal Protection

C1 Discretionary Protection

C2 Controlled Access Protection

B1 Labeled Security

B2 Structured Protection

B3 Security Domains

A1 Verified Protection

**Discretionary Protection (Categories C1, C2)** C1 and C2 systems provide basic controls and complete documentation for system installation and configuration.

**Discretionary Security Protection (C1)** A discretionary security protection system controls access by user IDs and/or groups. Although there are some controls in place that limit object access, **systems in this category provide only weak protection.**

**Controlled Access Protection (C2)** Controlled access protection systems are stronger than C1 systems. Users must be identified individually to gain access to objects. **C2 systems must also enforce media cleansing**. With **media cleansing**, any media that are reused by another user must first be thoroughly cleansed so that no **remnant of the previous** data remains available for inspection or use. Additionally, strict logon procedures must be enforced that restrict access for invalid or unauthorized users.

**Mandatory Protection (Categories B1, B2, B3)** Mandatory protection systems provide more security controls than category C or D systems. More granularity of control is mandated, so security administrators can apply specific controls that allow only very limited. sets of subject/object access. **This category of systems is based on the Bell-LaPadula model. Mandatory access is based on security labels.**

**Labeled Security (B1)** In a labeled security system, each subject and each object has a security label. A B1 system grants access by matching up the subject and object labels and comparing their permission compatibility. B1 systems support sufficient security to house classified data.

**Structured Protection (B2)** In addition to the requirement for security labels (as in B1 systems), B2 systems must ensure that **no covert channels exist.** **Operator and administrator functions are separated**, and process isolation is maintained. B2 systems are sufficient for classified data that requires more security functionality than a B1 system can deliver.

**Security Domains (B3)** Security domain systems **provide more secure functionality by further increasing the separation and isolation of unrelated processes**. Administration functions are clearly defined and separate from functions available to other users. The focus of B3 systems shifts to simplicity to reduce any exposure to vulnerabilities in unused or extra code. The secure state of B3 systems must also be addressed during the initial boot process. B3 systems are difficult to attack successfully and provide sufficient secure controls for very sensitive or secret data.

**Verified Protection (Category A1)** Verified protection systems are similar to B3 systems in the structure and controls they employ. The difference is in the development cycle. Each phase of the development cycle is controlled using formal methods. Each phase of the design is documented, evaluated, and verified before the next step is taken. This force extreme security consciousness during all steps of development and deployment and is the only way to formally guarantee strong system security.

A verified design system starts with a design document that states how the resulting system will satisfy the security policy. From there, each development step is evaluated in the context of the security policy. Functionality is crucial, but assurance becomes more important than in lower security categories. A1 systems represent the top level of security and are designed to handle top-secret data. Every step is documented and verified, from the design all the way through to delivery and installation.

**Red Book** Because the Orange Book applies only to stand-alone computers not attached to a network, and so many systems were used on networks (even in the 1980s), **the Red Book was developed to interpret the TCSEC in a networking context. In fact, the official title of the Red Book is Trusted Network Interpretation of the TCSEC so it could be considered an interpretation of the Orange Book with** a bent on networking. Quickly the Red Book became more relevant and important to system buyers and builders than the Orange Book. The following list includes a few other functions of the Red Book:

■ Rates confidentiality and integrity

■ Addresses communications integrity

■ Addresses denial of service protection

■ Addresses compromise (in other words, intrusion) protection and prevention

■ Is restricted to a limited class of networks that are labeled as “centralized networks with a single accreditation authority”

■ Uses only four rating levels: None, C1 (Minimum), C2 (Fair), and B2 (Good)

**Green Book** The Green Book, or the Department of Defense Password Management Guidelines , provides **password creation and management guidelines; it’s important for those who configure and manage trusted systems.**

5200.28-STD DoD Trusted Computer System Evaluation Criteria Orange Book

CSC-STD-002-85 DoD Password Management Guidelines Green Book

CSC-STD-003-85 Guidance for Applying TCSEC in Specific Environments Yellow Book

NCSC-TG-001 A Guide to Understanding Audit in Trusted Systems Tan Book

NCSC-TG-002 Trusted Product Evaluation: A Guide for Vendors Bright Blue Book

NCSC-TG-002- 85 PC Security Considerations Light Blue Book

NCSC-TG-003 A Guide to Understanding Discretionary Access Controls in Trusted Systems Neon Orange Book

NCSC-TG-004 Glossary of Computer Security Terms Aqua Book

NCSC-TG-005 Trusted Network Interpretation Red Book

NCSC-TG-006 A Guide to Understanding Configuration Management in Trusted Systems Amber Book

NCSC-TG-007 A Guide to Understanding Design Documentation in Trusted Systems Burgundy Book

NCSC-TG-008 A Guide to Understanding Trusted Distribution in Trusted Systems Lavender Book

NCSC-TG-009 Computer Security Subsystem Interpretation of the TCSEC Venice Blue Book

Major critiques of TCSEC; they help to explain why newer standards are now in use worldwide:

■ Although the TCSEC puts considerable emphasis on controlling user access to information, it doesn’t exercise control over what users do with information once access is granted. This can be a problem in military and commercial applications alike.

■ Given the origins of evaluation standards at the US Department of Defense, it’s understandable that the **TCSEC focuses its concerns entirely on confidentiality**, which assumes that controlling how users access data is of primary importance and that concerns about data accuracy or integrity are irrelevant. This doesn’t work in commercial environments where concerns about data accuracy and integrity can be more important than concerns about confidentiality.

■ Outside the evaluation standards’ own emphasis on access controls, the TCSEC does not carefully address the kinds of personnel, physical, and procedural policy matters or safeguards that must be exercised to fully implement security policy. They don’t deal much with how such matters can impact system security either.

■ The Orange Book, per se, doesn’t deal with networking issues (though the Red Book, developed later in 1987, does)

**ITSEC Classes and Required Assurance and Functionality**

The ITSEC represents an initial attempt to create security evaluation criteria in Europe. It was developed as an alternative to the TCSEC guidelines. The ITSEC guidelines evaluate the functionality and assurance of a system using separate ratings for each category. In this context, a system’s functionality is a measurement of the system’s utility value for users. The functionality rating of a system states how well the system performs all necessary functions based on its design and intended purpose. The assurance rating represents the degree of confidence that the system will work properly in a consistent manner.

**ITSEC refers to any system being evaluated as a target of evaluation (TOE). All ratings are expressed as TOE ratings in two categories. ITSEC uses two scales to rate** **functionality and assurance**

**The functionality of a system is rated from F-D through F-B3 (there is no F-A1). The assurance of a system is rated from E0 through E6. Most ITSEC ratings generally correspond with TCSEC ratings**

Differences between TCSEC and ITSEC are many and varied. The following are some of the most important differences between the two standards:

■ **Although the TCSEC concentrates almost exclusively on confidentiality, ITSEC addresses concerns about the loss of integrity and availability in addition to confidentiality, thereby covering all three elements so important to maintaining complete information security.**

■ ITSEC does not rely on the notion of a TCB, and it doesn’t require that a system’s security components be isolated within a TCB.

■ Unlike TCSEC, which required any changed systems to be reevaluated anew—be it for operating system upgrades, patches, or fixes; application upgrades or changes; and so forth—ITSEC includes coverage for maintaining targets of evaluation after such changes occur without requiring a new formal evaluation.

**Common Criteria**

The Common Criteria represents a more or less global effort that involves everybody who worked on TCSEC and ITSEC as well as other global players. Ultimately, it results in the ability to purchase CC-evaluated products (where CC, of course, stands for Common Criteria). The Common Criteria defines various levels of testing and confirmation of systems’ security capabilities, and the number of the level indicates what kind of testing and confirmation has been performed. Nevertheless, it’s wise to observe that even the highest CC ratings do not equate to a guarantee that such systems are completely secure or that they are entirely devoid of vulnerabilities or susceptibilities to exploit. **The Common Criteria was designed as a product evaluation model.**

**Recognition of Common Criteria**

Caveats and disclaimers aside, a document titled “**Arrangement on the Recognition of Common Criteria Certificates in the Field of IT Security” was signed by representatives from government organizations in Canada, France, Germany, the United** Kingdom, and the United States in 1998, making it an international standard. This document was converted by ISO into an official standard: **ISO 15408**, Evaluation Criteria for Information Technology Security. The objectives of the CC guidelines are as follows:

To add to buyers’ confidence in the security of evaluated, rated IT products

■ To eliminate duplicate evaluations (among other things, this means that if one country, agency, or validation organizations follows the CC in rating specific systems and configurations, others elsewhere need not repeat this work)

■ To keep making security evaluations and the certification process more cost effective and efficient

■ To make sure evaluations of IT products adhere to high and consistent standards

■ To promote evaluation and increase availability of evaluated, rated IT products

■ To evaluate the functionality (in other words, what the system does) and assurance (in other words, how much can you trust the system) of the TOE

The Common Criteria process is based on two key elements: **protection profiles and security targets**. **Protection profiles (PPs) specify for a product that is to be evaluated (the TOE) the security requirements and protections, which are considered the security desires or the “I want” from a customer. Security targets (STs) specify the claims of security from the vendor that are built into a TOE. STs are considered the implemented security measures or the “I will provide” from the vendor**

**In addition to offering security targets, vendors may offer packages of additional security features. A package is an intermediate grouping of security requirement components that can be added or removed from a TOE.**

The PP is compared to various STs from the selected vendor’s TOEs. The closest or best match is what the client purchases**. The client initially selects a vendor based on published or marketed evaluation assurance levels (EALs**) for currently available systems. Using Common Criteria to choose a vendor allows clients to request exactly what they need for security rather than having to use static fixed security levels. It also allows vendors more flexibility on what they design and create. A well-defined set of Common Criteria supports subjectivity and versatility, and it automatically adapts to changing technology and threat conditions. Furthermore, **the EALs provide a method for comparing vendor systems that is more standardized** (like the old TCSEC).

**Structure of the Common Criteria**

The CC guidelines are divided into three areas, as follows:

**Part 1** **Introduction and General Model describes the general concepts** and underlying model used to evaluate IT security and what’s involved in specifying targets of evaluation. It contains useful introductory and explanatory material for those unfamiliar with the workings of the security evaluation process or who need help reading and interpreting evaluation results.

**Part 2 Security Functional Requirements describes various functional requirements in terms of security audits, communications security, cryptographic support for security, user data protection, identification and authentication, security management, TOE security functions (TSFs), resource util**ization, system access, and trusted paths. Covers the complete range of security functions as envisioned in the CC evaluation process, with additional appendices (called annexes s) to explain each functional area.

**Part 3** Security Assurance covers assurance requirements for TOEs in the areas of configuration management, **delivery and operation, development, guidance documents, and life-cycle support plus assurance tests and vulnerability assessments**. Covers the complete range of security assurance checks and protects profiles as envisioned in the CC evaluation process, with information on evaluation assurance levels that describe how systems are designed, checked, and tested.

**EAL1 Functionally tested** Applies when some confidence in correct operation is required but where threats to security are not serious. This is of value when independent assurance that due care has been exercised in protecting personal information is necessary.

**EAL2 Structurally tested** Applies when delivery of design information and test results are in keeping with good commercial practices. This is of value when developers or users require low to moderate levels of independently assured security. IT is especially relevant when evaluating legacy systems.

**EAL3 Methodically tested and checked** Applies when security engineering begins at the design stage and is carried through without substantial subsequent alteration. This is of value when developers or users require a moderate level of independently assured security, including thorough investigation of TOE and its development.

**EAL4 Methodically designed, tested, and reviewed** Applies when rigorous, positive security engineering and good commercial development practices are used. This does not require substantial specialist knowledge, skills, or resources. It involves independent testing of all TOE security functions.

**EAL5 Semi-formally designed and tested** Uses rigorous security engineering and commercial development practices, including specialist security engineering techniques, for semi-formal testing. This applies when developers or users require a high level of independently assured security in a planned development approach, followed by rigorous development.

**EAL6 Semi-formally verified, designed, and tested** Uses direct, rigorous security engineering techniques at all phases of design, development, and testing to produce a premium TOE. This applies when TOEs for high-risk situations are needed, where the value of protected assets justifies additional cost. Extensive testing reduces risks of penetration, probability of cover channels, and vulnerability to attack.

**EAL7 Formally verified, designed, and tested** Used only for highest-risk situations or where highvalue assets are involved. This is limited to TOEs where tightly focused security functionality is subject to extensive formal analysis and testing.

The CC guidelines also do not address administrative issues outside the specific purview of security. As with other evaluation criteria, the CC guidelines do not include evaluation of security in situ —that is, they do not address controls related to personnel, organizational practices and procedures, or physical security. Likewise, controls over electromagnetic emissions are not addressed, nor are the criteria for rating the strength of cryptographic algorithms explicitly laid out. Nevertheless, the CC guidelines represent some of the best techniques whereby systems may be rated for security

TCSEC ITSEC CC description

D F-D+E0 EAL0, EAL1 Minimal/no protection

C1 F-C1+E1 EAL2 Discretionary security mechanisms

C2 F-C2+E2 EAL3 Controlled access protection

B1 F-B1+E3 EAL4 Labeled security protection

B2 F-B2+E4 EAL5 Structured security protection

B3 F-B3+E5 EAL6 Security domains

A1 F-B3+E6 EAL7 Verified security design

**Certification**

The first phase in a total evaluation process is certification . Certification is the comprehensive evaluation of the technical and nontechnical security features of an IT system. and other safeguards made in support of the accreditation process to establish the extent to which a particular design and implementation meets a set of specified security requirements. System certification is the technical evaluation of each part of a computer system to assess its concordance with security standards. First, you must choose evaluation criteria. Once you select criteria to use, you analyze each system component to determine whether it satisfies the desired security goals. The certification analysis includes testing the system’s hardware, software, and configuration. All controls are evaluated during this phase, including administrative, technical, and physical controls.

**Accreditation**

In the certification phase, you test and document the security capabilities of a system in a specific configuration. With this information in hand, the management of an organization compares the capabilities of a system to the needs of the organization. It is imperative that the security policy clearly states the requirements of a security system. Management reviews the certification information and decides whether the system satisfies the security needs of the organization. If management decides the certification of the system satisfies their needs, the system is accredited. Accreditation i d s the formal declaration by the designated approving authority (DAA) that an IT system is approved to operate in a particular security mode using a prescribed set of safeguards at an acceptable level of risk. Once accreditation is performed, management can formally accept the adequacy of the overall security performance of an evaluated system.

**Certification and Accreditation Systems**

Two government standards are currently in place for the certification and accreditation of computing systems. The current DoD standard is Risk Management Framework(RMF) which recently replaced DoD Information Assurance Certification and Accreditation Process (DIACAP), which itself replaced the Defense Information Technology Security Certification and Accreditation Process (DITSCAP). The standard for all other US government executive branch departments, agencies, and their contractors and consultants is the Committee on National Security Systems (CNSS) Policy (CNSSP) which replaced National Information Assurance Certification and Accreditation Process (NIACAP). However, the CISSP may refer to either the current standards or the previous ones. Both of these processes are divided into four phases:

**Phase 1: Definition** Involves the assignment of appropriate project personnel; documentation of the mission need; and registration, negotiation, and creation of a System Security Authorization Agreement (SSAA) that guides the entire certification and accreditation process

**Phase 2: Verification** Includes refinement of the SSAA, systems development activities, and a certification analysis

**Phase 3: Validation** Includes further refinement of the SSAA, certification evaluation of the integrated system, development of a recommendation to the DAA, and the DAA’s accreditation decision

**Phase 4: Post Accreditation** Includes maintenance of the SSAA, system operation, change management, and compliance validation

The NIACAP process, administered by the Information Systems Security Organization of the National Security Agency, outlines three types of accreditation that may be granted.

The definitions of these types of accreditation (from National Security Telecommunications and Information Systems Security Instruction 1000) are as follows:

■ For a system accreditation, a major application or general support system is evaluated.

■ For a site accreditation, the applications and systems at a specific, self-contained location are evaluated.

■ For a type accreditation, an application or system that is distributed to a number of different locations is evaluated.

**Memory Protection**

Memory protection is a core security component that must be designed and implemented into an operating system. It must be enforced regardless of the programs executing in the system. Otherwise instability, violation of integrity, denial of service, and disclosure are likely results. Memory protection is used to prevent an active process from interacting with an area of memory that was not specifically assigned or allocated to it.

**Virtualization**

Virtualization technology is used to host one or more operating systems within the memory of a single host computer. This mechanism allows virtually any OS to operate on any hardware. It also allows multiple OSs to work simultaneously on the same hardware. Common examples include VMware, Microsoft’s Virtual PC, Microsoft Virtual Server, Hyper-V with Windows Server, Oracle’s VirtualBox, XenServer, and Parallels Desktop for Mac.

**Trusted Platform Module**

The Trusted Platform Module (TPM) is both a specification for a crypto processor chip on a mainboard and the general name for implementation of the specification. A TPM chip is used to store and process cryptographic keys for the purposes of a hardware supported/implemented hard drive encryption system. Generally, a hardware implementation, rather than a software-only implementation of hard drive encryption, is considered to be more secure.

When TPM-based whole-disk encryption is in use, the user/operator must supply a password or physical USB token device to the computer to authenticate and allow the TPM chip to release the hard drive encryption keys into memory. While this seems similar to a software implementation, the key difference is that if the hard drive is removed from its original system, it cannot be decrypted. Only with the original TPM chip can an encryption be decrypted and accessed. With software-only hard drive encryption, the hard drive can be moved to a different computer without any access or use limitations.

**A hardware security module (HSM) is a crypto processor used to manage/store digital encryption keys, accelerate crypto operations, support faster digital signatures, and improve authentication. An HSM is often an add-on adapter or peripheral or can be a TCP/IP network device. HSMs include tamper protection to prevent their misuse even if physical access is gained by an attacker. A TPM is just one example of an HSM.**

HSMs provide an accelerated solution for large (2,048+ bit) asymmetric encryption calculations and a secure vault for key storage. Many certificate authority systems use HSMs to store certificates; ATM and POS bank terminals often employ proprietary HSMs; hardware SSL accelerators can include HSM support; and DNSSEC-compliant DNS servers use HSM for key and zone fi le storage.

**Interfaces**

A constrained or restricted interface is implemented within an application to restrict what users can do or see based on their privileges. Users with full privileges have access to all the capabilities of the application. Users with restricted privileges have limited access. Applications constrain the interface using different methods. A common method is to hide the capability if the user doesn’t have permissions to use it. Commands might be available to administrators via a menu or by right-clicking an item, but if a regular user doesn’t have permissions, the command does not appear. Other times, the command is shown but is dimmed or disabled. The regular user can see it but will not be able to use it. The purpose of a constrained interface is to limit or restrict the actions of both authorized and unauthorized users. The use of such an interface is a practical implementation of the Clark-Wilson model of security.

**Fault Tolerance**

Fault tolerance is the ability of a system to suffer a fault but continue to operate. Fault tolerance is achieved by adding redundant components such as additional disks within a redundant array of inexpensive disks (RAID) array, or additional servers within a failover clustered configuration. Fault tolerance is an essential element of security design. It is also considered part of avoiding single points of failure and the implementation of redundancy. For more details on fault tolerance, redundant servers, RAID, and failover solutions

.