identity.
• A registration authority (RA) validates the user’s identity and then sends the request
for a certificate to the CA. The RA cannot generate certificates.
• A one-way function is a mathematical function that is easier to compute in one direction than in the opposite direction.
• RSA is based on a one-way function that factors large numbers into prime numbers.
Only the private key knows how to use the trapdoor and how to decrypt messages that were encrypted with the corresponding public key.
• Hashing algorithms provide data integrity only.
• When a hash algorithm is applied to a message, it produces a message digest, and
this value is signed with a private key to produce a digital signature.
• Some examples of hashing algorithms include SHA-1, SHA-2, SHA-3, and MD5.
• SHA produces a 160-bit hash value and is used in DSS.

• A birthday attack is an attack on hashing functions through brute force. The attacker tries to create two messages with the same hashing value.
• A one-time pad uses a pad with random values that are XORed against the message to produce ciphertext. The pad is at least as long as the message itself and is used once and then discarded.
• A digital signature is the result of a user signing a hash value with a private key. It
provides authentication, data integrity, and nonrepudiation. The act of signing is
the actual encryption of the value with the private key.
• Key management is one of the most challenging pieces of cryptography. It pertains
to creating, maintaining, distributing, and destroying cryptographic keys.
• Brute-force attacks against cryptosystems systematically try all possible keys against given ciphertext in hopes of guessing the key that was used.
• Ciphertext-only attacks against cryptosystems involve analyzing the ciphertext of one or more messages encrypted with the same algorithm and key in order to discover the key that was used.
• In a known-plaintext attack, the attacker has the plaintext and corresponding ciphertext of one or more messages and wants to discover the key that was used.
• A chosen-plaintext attack is like a known-plaintext attack but the attacker chooses the plaintext that gets encrypted to see the corresponding ciphertext.
• A chosen-ciphertext attack is like a chosen-plaintext attack except that the attacker
chooses the ciphertext and then gets to see the corresponding decrypted plaintext.
• A frequency analysis, also known as a statistical attack, identifies statistically
significant patterns in the ciphertext generated by a cryptosystem.
• Implementation attacks are the techniques used to exploit defects in the implementation of a cryptosystem.
• Side-channel attacks analyze changes in the environment around a cryptosystem in an attempt to infer an encryption key whose processing causes those changes.
• Timing attacks are side-channel attacks that use time measurements to

determine
the inner workings, states, and even data flows within a cryptosystem.
• Fault injection attacks attempt to cause errors in a cryptosystem in an attempt to
recover or infer the encryption key.
• In man-in-the-middle (MitM) attacks, threat actors intercept an outbound secure
connection request from clients and relay their own requests to the intended
servers, terminating both and acting as a proxy.
• Pass the hash is a type of attack against Microsoft Windows Active Directory in
which the attacker resubmits cached authentication tokens to gain illicit access to
resources.
• Ransomware is a type of malware that encrypts victims' files and holds them
ransom until a payment is made to an account controlled by the attacker.

Questions
Please remember that these questions are formatted and asked in a certain way for a
reason. Keep in mind that the CISSP exam is asking questions at a conceptual level.
Questions may not always have the perfect answer, and the candidate is advised against
always looking for the perfect answer. Instead, the candidate should look for the best
answer in the list.
1. What is the goal of cryptanalysis?
A. To determine the strength of an algorithm
B. To increase the substitution functions in a cryptographic algorithm
C. To decrease the transposition functions in a cryptographic algorithm

2. Why has the frequency of successful brute-force attacks increased?
A. The use of permutations and transpositions in algorithms has increased.
B. As algorithms get stronger, they get less complex, and thus more susceptible

to attacks.
C. Processor speed and power have increased.
D. Key length reduces over time.
3. Which of the following is not a property or characteristic of a one-way hash function?
A. It converts a message of arbitrary length into a value of fixed length.
B. Given the digest value, finding the corresponding message should be computationally infeasible.
C. Deriving the same digest from two different messages should be impossible or rare.
D. It converts a message of fixed length to an arbitrary length value.
4. What would indicate that a message had been modified?
A. The public key has been altered.
B. The private key has been altered.

C. The message digest has been altered.
D. The message has been encrypted properly.
5. Which of the following is a U.S. federal government algorithm developed for creating secure message digests?
A. Data Encryption Algorithm
B. Digital Signature Standard
C. Secure Hash Algorithm
D. Data Signature Algorithm

PART III

D. To determine the permutations used

6. What is an advantage of RSA over DSA?
A. It can provide digital signature and encryption functionality.
B. It uses fewer resources and encrypts faster because it uses symmetric keys.
C. It is a block cipher rather than a stream cipher.
D. It employs a one-time encryption pad.

7. What is used to create a digital signature?
A. The receiver’s private key
B. The sender’s public key
C. The sender’s private key
D. The receiver’s public key

8. Which of the following best describes a digital signature?
A. A method of transferring a handwritten signature to an electronic document
B. A method to encrypt confidential information
C. A method to provide an electronic signature and encryption
D. A method to let the receiver of the message prove the source and integrity of

a message
9. Why would a certificate authority revoke a certificate?
A. If the user’s public key has become compromised
B. If the user changed over to using the PEM model that uses a web of trust
C. If the user’s private key has become compromised
D. If the user moved to a new location
10. Which of the following best describes a certificate authority?
A. An organization that issues private keys and the corresponding algorithms
B. An organization that validates encryption processes
C. An organization that verifies encryption keys
D. An organization that issues certificates
11. Which of the following is a true statement pertaining to data encryption when it
is used to protect data?
A. It verifies the integrity and accuracy of the data.
B. It requires careful key management.
C. It does not require much system overhead in resources.
D. It requires keys to be escrowed.

381
12. What is the definition of an algorithm’s work factor?
A. The time it takes to encrypt and decrypt the same plaintext
B. The time it takes to break the encryption
C. The time it takes to implement 16 rounds of computation
D. The time it takes to apply substitution functions

13. What is the primary purpose of using one-way hashing on user passwords?
A. It minimizes the amount of primary and secondary storage needed to store

Answers
1. A. Cryptanalysis is the process of trying to reverse-engineer a cryptosystem, with
the possible goal of uncovering the key used. Once this key is uncovered, all other
messages encrypted with this key can be accessed. Cryptanalysis is carried out by
the white hats to test the strength of the algorithm.
2. C. A brute-force attack is resource intensive. It tries all values until the correct
one is obtained. As computers have more powerful processors added to them,
attackers can carry out more powerful brute-force attacks.

PART III

passwords.
B. It prevents anyone from reading passwords in plaintext.
C. It avoids excessive processing required by an asymmetric algorithm.
D. It prevents replay attacks.
14. Which of the following is based on the fact that it is hard to factor large numbers
into two original prime numbers?
A. ECC
B. RSA
C. SHA
D. MD5
15. What is the name given to attacks that involve analyzing changes in the
environment around a cryptosystem to infer the encryption key whose processing
causes those changes?
A. Side-channel attack
B. Timing attack
C. Implementation attack
D. Fault injection attack

382
3. D. A hashing algorithm will take a string of variable length (the message can be
any size) and compute a fixed-length value. The fixed-length value is the message
digest. The MD family creates the fixed-length value of 128 bits, and SHA creates

one of 160 bits.
4. C. Hashing algorithms generate message digests to detect whether modification has
taken place. The sender and receiver independently generate their own digests, and
the receiver compares these values. If they differ, the receiver knows the message has
been altered.
5. C. SHA was created to generate secure message digests. Digital Signature Standard
(DSS) is the standard to create digital signatures, which dictates that SHA must
be used. DSS also outlines the digital signature algorithms that can be used with
SHA: RSA, DSA, and ECDSA.
6. A. RSA can be used for data encryption, key exchange, and digital signatures. DSA
can be used only for digital signatures.
7. C. A digital signature is a message digest that has been encrypted with the sender's
private key. A sender, or anyone else, should never have access to the receiver's
private key.
8. D. A digital signature provides authentication (knowing who really sent the
message), integrity (because a hashing algorithm is involved), and nonrepudiation
(the sender cannot deny sending the message).
9. C. The reason a certificate is revoked is to warn others who use that person's public
key that they should no longer trust the public key because, for some reason, that
public key is no longer bound to that particular individual's identity. This could
be because an employee left the company or changed his name and needed a new
certificate, but most likely it is because the person's private key was compromised.
10. D. A registration authority (RA) accepts a person's request for a certificate and
verifies that person's identity. Then the RA sends this request to a certificate
authority (CA), which generates and maintains the certificate.
11. B. Data encryption always requires careful key management. Most algorithms are
so strong today that it is much easier to go after key management than to launch
a brute-force attack. Hashing algorithms are used for data integrity, encryption
does require a good amount of resources, and keys do not have to be escrowed
for encryption.
12. B. The work factor of a cryptosystem is the amount of time and resources necessary
to break the cryptosystem or its encryption process. The goal is to make the work
factor so high that an attacker could not be successful in breaking the algorithm or
cryptosystem.
13. B. Passwords are usually run through a one-way hashing algorithm so that the
actual

password is not transmitted across the network or stored on a system in plaintext.
This greatly reduces the risk of an attacker being able to obtain the actual password.

↟Chapter 8: Cryptology

383
14. B. The RSA algorithm’s security is based on the difficulty of factoring large numbers
into their original prime numbers. This is a one-way function. Calculating the product is easier than identifying the prime numbers used to generate that product.
15. A. Side-channel attack is the best answer. The question could also describe a
timing attack, which is a kind of side-channel attack, but because there is no specific mention of timing, this option is not the best answer. An argument could also be made for this being a fault injection attack but, again, there is no
specific mention of the attacker deliberately trying to cause errors. This question
is representative of the harder questions in the CISSP exam, which provide one or
more answers that could be correct but are not the best ones.

PART III

↟This page intentionally left blank

↟CHAPTER

Security Architectures
This chapter presents the following:
• Threat modeling
• Security architecture design principles
• Security models
• Addressing security requirements
• Security capabilities of information systems

Security is a process, not a product.
—Bruce Schneier
Having discussed the various information system architectures (in Chapter 7) and cryptology (in Chapter 8), our next step is to put these together as we develop an enterprise
security architecture. We already discussed the various frameworks for this in Chapter 4,
so what we need to do now is to apply tried and true principles, models, and system
capabilities. Before we do this, however, we will explore threat modeling in a fair amount
of detail, since it informs everything else we do.

Threat Modeling
Before we can develop effective defenses, it is imperative to understand the

assets that
we value, as well as the threats against which we are protecting them. Though multiple
definitions exist for the term, for the purposes of our discussion we define threat modeling
as the process of describing probable adverse effects on our assets caused by specific threat
sources. That’s quite a mouthful, so let’s break it down.
When we build a model of the threats we face, we want to ground them in reality, so it
is important to only consider dangers that are reasonably likely to occur. To do otherwise
would dilute our limited resources to the point of making us unable to properly defend
ourselves. Next, we want to focus our work on the potential impact of those threats to
organizational assets or, in other words, to things and people that are of value to the
organization. Lastly, the model needs to specify the threat sources if we are to develop
effective means to thwart them. We must understand their capabilities and motivations
if we are to make sense of their actions.

385

9

↟CISSP All-in-One Exam Guide

386

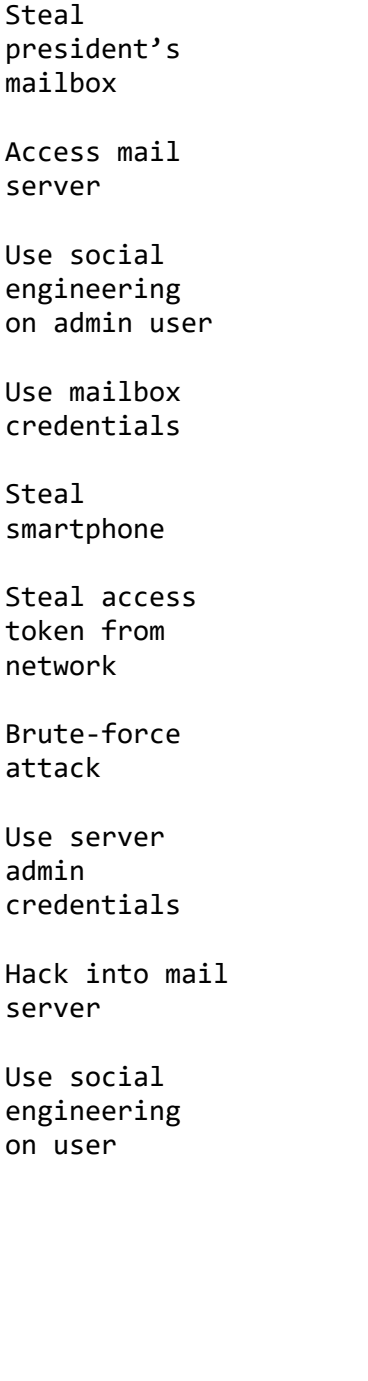
Attack Trees
An attack tree is a graph showing how individual actions by attackers can be chained
together to achieve their goals. This methodology is based on the observation
that, typically, there are multiple ways to accomplish a given objective. For
example, if a disgruntled employee wanted to steal the contents of the president’s mailbox, she could
accomplish this by either accessing the e-mail server, obtaining the password, or stealing
the president’s smartphone. Accessing the e-mail server could be accomplished by using
administrative credentials or by hacking in. To get the credentials, she could use brute
force or social engineering. The options available to the attacker create the branches in
the tree, an example of which is shown in Figure 9-1. Each of the leaf nodes represents
a specific condition that must be met in order for the parent node to be effective. For
instance, to effectively obtain the mailbox credentials, the disgruntled employee could
have stolen a network access token. Given that the employee has met the

condition of having the credentials, she would then be able to steal the contents of the president's mailbox. A successful attack, then, is one in which the attacker traverses from a leaf node all the way to the root node at the top of the tree, which represents the ultimate objective.

NOTE The terms "attack chain" and "kill chain" are commonly used. They refer to a specific type of attack tree that has no branches and simply proceeds from one stage or action to the next. The attack tree is much more expressive in that it shows many ways in which an attacker can accomplish each objective.

Reduction Analysis

The generation of attack trees for an organization usually requires a large investment of resources. Each vulnerability-threat-attack triad can be described in detail using an attack tree, so you end up with as many trees as you do triads. To defeat each of the attacks



Steal access
token from
the network

Brute-force
attack

A given technique can be used by a
threat source in multiple attacks.

Figure 9-1 A simplified attack tree

STRIDE
STRIDE is a threat modeling framework that evaluates a system’s design using flow diagrams, system entities, and events related to a system. STRIDE is not an acronym, but a mnemonic for the six categories of security threats outlined in Table 9-1. STRIDE was developed in 1999 by Microsoft as a tool to help secure systems as they were being developed. STRIDE is among the most widely used threat-modeling frameworks and is suitable for application to logical and physical systems alike.

The Lockheed Martin Cyber Kill Chain
Threat modeling is really nothing new. It probably has its oldest roots in military operations, where it is used to anticipate the intent and actions of an enemy and then develop a plan to get inside their decision loop and defeat them. The term kill chain evolved to describe the process of identifying a target, determining the best way to engage it, amassing the required forces against it, engaging it, and destroying it. In 2011, Lockheed Martin released a paper defining a Cyber Kill Chain based on this methodology. It identifies the steps that threat actors generally must complete to achieve their objectives. This model specifies seven distinct stages of a cyberattack:
1. Reconnaissance The attacker selects a target, researches it, and attempts to identify vulnerabilities in the target network.
2. Weaponization The attacker either adapts an existing remote access malware weapon or creates a new one, tailored to one or more vulnerabilities identified in the previous step.

you identify, you would typically need a control or countermeasure at each leaf node.

Since one attack generates many leaf nodes, this has a multiplicative effect that could make it very difficult to justify the whole exercise. However, attack trees lend themselves to a methodology known as reduction analysis.

There are two aspects of reduction analysis in the context of threat modeling: one aspect is to reduce the number of attacks we have to consider, and the other is to reduce the threat posed by the attacks. The first aspect is evidenced by the commonalities in the example shown in Figure 9-1. To satisfy the conditions for logging into the mail server or the user's mailbox, an attacker can use the exact same three techniques. This means we can reduce the number of conditions we need to mitigate by finding these commonalities. When you consider that these three sample conditions apply to a variety of other attacks, you realize that we can very quickly cull the number of conditions to a manageable number.

The second aspect of reduction analysis is the identification of ways to mitigate or negate the attacks we've identified. This is where the use of attack trees can really benefit us. Recall that each tree has only one root but many leaves and internal nodes. The closer you are to the root when you implement a mitigation technique, the more leaf conditions you will defeat with that one control. This allows you to easily identify the most effective techniques to protect your entire organization. These techniques are typically called controls or countermeasures.

Threat

Property Affected

Spoofing

Definition

Example

Authentication

Impersonating someone or something else

An outside sender
pretending to be an HR
employee in an e-mail

Tampering

Integrity

Modifying data on disk, in
memory, or elsewhere

A program modifying
the contents of a critical
system file

Repudiation

Nonrepudiation

Claiming to have not
performed an action or
to have no knowledge of
who performed it

A user claiming that she did
not receive a request

Information
disclosure

Confidentiality

Exposing information to
parties not authorized to
see it

An analyst accidentally
revealing the inner details
of the network to outside
parties

Denial of
service

Availability

Denying or degrading
service to legitimate users by
exhausting resources needed
for a service

A botnet flooding a website
with thousands of requests a

second, causing it to crash

Elevation of privilege

Authorization

Gaining capabilities without the proper authorization to do so

A user bypassing local restrictions to gain administrative access to a workstation

Table 9-1 STRIDE Threat Categories

3. Delivery The attacker transmits the weapon to the target (e.g., via e-mail attachments, links to malicious websites, or USB drives).
4. Exploitation The malware weapon triggers, which takes action on the target to exploit one or more vulnerabilities and compromise the host.
5. Installation The malware weapon installs an access point (e.g., backdoor) usable by the attacker.
6. Command and Control The malware enables the attacker to have “hands on the keyboard” persistent access to the target network.
7. Actions on Objective The attacker takes actions to achieve their goals, such as data exfiltration, data destruction, or encryption for ransom.

One of Lockheed Martin’s key goals of developing this model was to allow defenders to map defensive measures to each stage and ensure that they have sufficient coverage to detect, deny, disrupt, degrade, deceive, or contain the attack. The earlier in the kill chain this is done, the better, because the adversary will not have attained their objective yet.
Another key idea in the model is that of identifying indicators of adversarial activity at each stage of a cyberattack, which would allow defenders to detect the activities but also

determine whether the defensive measures at a particular stage were effective. Though the Cyber Kill Chain is a high-level framework, it is one of the most commonly used ones for modeling threat activities.

The MITRE ATT&CK Framework

Why Bother with Threat Modeling

A scientific model is a simplified representation of something that is difficult to understand. The model is built in a way that makes certain parts of the subject easier to observe and analyze. We use these models to study complex phenomena like the spread of disease, global financial markets, and, of course, cybersecurity threats. Threat modeling allows us to simplify some of the activities of our adversaries so we can drill into the parts that really matter to us as defenders. There are just too many threat actors doing too many discrete things in too many places for us to study each in detail. Instead, we look for likely attackers and patterns that allow us to mitigate a bunch of different attacks by defeating the techniques that they all have in common.

Let's continue our previous example of the Denis Trojan using DNS tunneling. Should we care about it? That depends on what organization we belong to. We would probably care if we happen to be part of a media and human rights organizations in or around Vietnam, where OceanLotus/APT32 appears to be focused. This is the power of threat modeling: it allows us to focus keenly on specific actors and specific techniques based on who and where our organization is.

A typical threat modeling effort would start by identifying threat actors that are likelier to target our organization. This can be general classes of actors, such as opportunistic ransomware gangs, or more specific ones, like OceanLotus. Where do we get this information? Maybe we read the news and look for attackers that are targeting organizations such as ours. Or maybe we are subscribed to threat intelligence services

PART III

The MITRE Corporation developed a framework of adversarial tactics, techniques, and common knowledge called ATT&CK as a comprehensive matrix of tactics and techniques used by threat actors. It is a widely used tool to construct models of complex campaigns and operations using reusable common components. Like the Cyber Kill Chain, ATT&CK breaks down actions into (generally) sequential groupings called tactics, which can be mapped to those in the Lockheed Martin model. Each of the 14 tactics contains a number of techniques by which the adversary may achieve a particular purpose. The techniques, in turn, contain specific sub-techniques used by named threat actors.

For example, the eleventh tactic (T0011), Command and Control, describes

techniques used by adversaries when trying to communicate with compromised systems
to control them. One of these techniques (T1071) deals with the use of application layer
protocols to establish communications in a discrete manner. One of the sub-techniques
(T1071.004) involves the use of the Domain Name System (DNS) to send and receive
messages covertly. This sub-technique, in turn, contains examples of procedures used by
various known threat actors, such as OceanLotus (also known as APT32) using a DNS
tunneling for command and control on its Denis Trojan.

that specifically look for those who are coming after us. Either way, we start by answering
a series of questions:

- Why might someone want to target our organization? (Motive)
- How could they go about accomplishing their objectives? (Means)
- When and where would they attack us? (Opportunity)

The answer to the first question comes from our asset inventory. What do we have
that is of value to others? Intellectual property would be valuable to competitors.
Cybercriminals would like to get any financial data. Foreign governments would be
after national security information. Once we know what kind of threat actors would
be interested in us, we can study how they go about attacking organizations like ours.
Going back to our example, suppose we are in an organization that might be of interest
to OceanLotus. We can research any available open or private sources to learn about
their tactics, techniques, and procedures (TTPs). The MITRE ATT&CK framework,
for instance, lists over 40 techniques and at least 13 tools used by OceanLotus. Knowing
their motivation and means, we can examine our systems and determine where they
could use those techniques. This is the power of threat modeling: it allows us to focus
our attention on understanding the threats that are likeliest to be of concern to our
organizations. Out of the countless TTPs used by adversaries around the world, we can
focus our attention on those that matter most to us.

Secure Design Principles

Understanding our threats is foundational to building secure architectures, but so is
applying the collective wisdom developed by the security community. This wisdom exists
in certain secure design principles that are widely recognized as best

practices. The sections that follow address the secure design principles that you should know for the CISSP
exam. Think of them as a solid starting point, not as an all-inclusive list.
EXAM TIP You should be able to describe each of the 11 secure design principles (which include the previously covered threat modeling) in some detail and recognize when they are (and are not) being followed in a given scenario.

Defense in Depth

One of the bedrock principles in designing security architectures is the assumption that
our jobs are not to determine whether our systems can be compromised, but rather to
prepare for the inevitable reality that this will happen. Consequently, we want to provide
defense in depth, which is the coordinated use of multiple security controls in a layered
approach, as shown in Figure 9-2. A multilayered defense system reduces the probability of successful penetration and compromise because an attacker would have to get
through several different types of protection mechanisms before she gained access to the

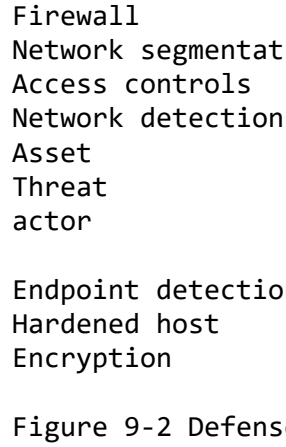


Figure 9-2 Defense in depth

- Perimeter fencing
- External lights
- Locked external and internal doors
- Security guard
- Security cameras

These measures would force the attacker, in many cases, to conduct the attack through
the network. Technical controls would then have to provide adequate protection to the
various assets in the company. Like their physical counterparts, you can think of these
technical controls as creating concentric circles of protection around your assets. For
example, as shown in Figure 9-2, you could have a perimeter firewall on the

outer ring. If the attacker circumvents this, she'd have to navigate through a segmented network with strictly enforced access controls everywhere. Beyond this she'd have to defeat network detection and response (NDR) systems that can detect and stop the attack as it moves across your network. Deeper still, there are endpoint detection and response (EDR) systems that are deployed on hardened hosts. Finally, all data is encrypted, which makes exfiltrating anything useful out of the company's environment more difficult.

Physical and technical controls are integrated and augmented by administrative controls such as policies, procedures, and standards. The types of controls that are actually implemented must map to the threats the company faces, and the number of layers that are put into place must map to the sensitivity of the asset. The rule of thumb is the more sensitive the asset, the more layers of protection that must be put into place.

critical assets. She may succeed at penetrating the perimeter and may get a foothold in the environment, but security controls are arranged in a way that will slow her down, improve the odds of detecting her, and provide means for defeating the attack or quickly recovering from it.

A well-designed security architecture considers the interplay of physical, technical, and administrative controls. For example, Company A can protect its facilities by using physical security controls such as the following:

Zero Trust

The principle of defense in depth is certainly useful, but some people may think it suggests that attacks always follow the pattern of external threat actors sequentially penetrating each defensive circle until they get to the asset being protected. In reality, attacks can come from any direction (even internally) and proceed nonlinearly. Some adversaries lay dormant in our networks for days, weeks, or even months waiting to complete their attacks from the inside. This reality, compounded by the threat of malicious or just careless insiders, has led many security professionals to a place of healthy paranoia.

A zero trust model is one in which every entity is considered hostile until

proven
otherwise. It considers trust as a vulnerability and tries to reduce or eliminate it in order
to make it harder for threat actors (external or internal) to accomplish their objectives.
If defense in depth looks at security from the outside in, zero trust architectures are built
from the inside out. These are not mutually exclusive approaches, however. It is best to
incorporate both principles as we design our systems.
The catch is that it is very hard to implement a zero trust model throughout an enterprise environment because it would hinder productivity and efficiency. For this
reason, this approach is most often focused only on a relatively small group of critical
assets, access to which defines a “protect surface,” which is where the tightest controls
are placed.

Trust But Verify
Another principle of designing secure architectures is trust but verify, which basically
means that, even when an entity and its behaviors are trusted, we should double-check
both. It could seem that this is an alternative to (or even incompatible with) the zero
trust model, but that is not necessarily true. You could, for instance, take a zero trust
approach to defending your most critical assets, while allowing certain trust (that you
verify) elsewhere. Of course, the zero trust assets would also be verified, and frequently.
In this manner, the two approaches can coexist happily. On the other hand, you could
take a hardline approach to one or the other and build your entire security architecture
around it. It really depends on your organization and its environment.
At the heart of the trust but verify principle is the implementation of mechanisms that
allow you to audit everything that happens on your systems. How else could you verify
them, right? But what is equally important is the development of processes by which you
pay the right amount of attention to auditing different elements of your environment.
You won’t have the ability to examine everything all the time so you have to figure out
how you’ll be able to analyze some things most of the time, and most things some of the
time. The risk is that we leave some parts of our environment unexamined and therefore
create a safe haven for attackers. Procedures matter just as much as technology.

Shared Responsibility

While the previous three principles (defense in depth, zero trust, and trust but verify)
can apply equally well to traditional, cloud, and hybrid environments, cloud computing adds a bit of complexity in terms of defensive roles. Shared responsibility refers to the

393
You manage
User Data

User Data

Applications

Applications

Applications

Services

Services

Services

Operating System

Operating System

Operating System

Hypervisor

Hypervisor

Hypervisor

Storage

Storage

Storage

Networking

Networking

Networking

Hardware

Hardware

Hardware

Facilities

Facilities

Facilities

SaaS

PaaS

IaaS

Figure 9-3 Shared responsibility in different cloud computing services

situation in which a service provider is responsible for certain security controls, while the customer is responsible for others. Typically, the service provider is responsible for security of the “thing” that they are providing. Think back to the three predominant cloud computing models we discussed in Chapter 7: Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS). Figure 9-3 shows the breakdown of responsibilities in each model.

The figure is, of course, a generalization. You should carefully read the fine print in the service agreements and discuss with your provider exactly what they will be doing for you and what they assume you’ll do for yourself. Then, you need to periodically ensure that these parameters have not changed due to changes in the environment, a revised agreement, or new features offered by your provider. Many cloud compromises can be traced to organizations not understanding they were responsible for providing a certain aspect of security.

Separation of Duties

Regardless of your physical or logical architecture, people will be responsible for performing various business, infrastructure, and security functions based on their roles within the organization. It is important to consider the human element in designing our security architectures. Granting access rights to our teammates should be based on the level of trust the organization has in them and in their need to know and do certain things. Just because a company completely trusts Joyce with its files and resources does not mean she fulfills the need-to-know criteria to access the company’s tax returns and profit margins.

If Maynard fulfills the need-to-know criteria to access employees’ work histories, it does not mean the company trusts him to access all of the company’s other files. These issues

must be identified and integrated into the access criteria. The different access criteria can be enforced by roles, groups, location, time, and transaction types.

Using roles is an efficient way to assign rights to a type of user who performs a certain task. This role is based on a job assignment or function. If there is a position within a company for a person to audit transactions and audit logs, the role this person fills would only need a read function to those types of files. If several users require the same type of access to information and resources, putting them into a group and then assigning rights and permissions to that group is easier to manage than assigning rights and permissions to each and every individual separately. If a specific printer is available only to the accounting group, when a user attempts to print to it, the group membership of the user will be checked to see if she is indeed in the accounting group.

Transaction-type restrictions can be used to further control what data is accessed during certain types of functions and what commands can be carried out on the data.

A purchasing agent in a company may be able to make purchases of up to $2,000, but would need a supervisor’s approval to buy more expensive items. The supervisor, conversely, might not be allowed to make any purchases at all, simply approve those submitted by a subordinate. This is an example of the principle of separation of duties (SoD), in which important functions are divided among multiple individuals to ensure that no one person has the ability to intentionally or accidentally cause serious losses to the organization.

Least Privilege

A related principle, least privilege, states that people are granted exactly the access and authority that they require to do their jobs, and nothing more. The purchasing agent’s

supervisor from the example in the previous section is not really expected to swipe his corporate credit card as part of his daily duties, so why should he even have a card? His purchasing agent is expected to make small purchases on a regular basis, but anything over $2,000 would be fairly rare, so why have a purchase limit any higher than that?

The need-to-know principle is similar to the least-privilege principle. It is based on the concept that individuals should be given access only to the information they absolutely require in order to perform their job duties. Giving any more rights to a user just asks for headaches and the possibility of that user abusing the permissions assigned to him. An administrator wants to give a user the least amount of privileges she can, but just enough for that user to be productive when carrying out tasks. Management decides what a user needs to know, or what access rights are necessary, and the administrator configures the access control mechanisms to allow this user to have that level of access and no more, and thus the least privilege.

For example, if management has decided that Dan, the intern, needs to know where the files he needs to copy are located and needs to be able to print them, this fulfills Dan's need-to-know criteria. Now, an administrator could give Dan full control of all the files he needs to copy, but that would not be practicing the least-privilege principle. The administrator should restrict Dan's rights and permissions to only allow him to read and print the necessary files, and no more. Besides, if Dan accidentally deletes all the files on

the whole file server, who do you think management will hold ultimately responsible?

Yep, the administrator.

It is important to understand that it is management's job to determine the security requirements of individuals and how access is authorized. The security administrator configures the security mechanisms to fulfill these requirements, but it is not her job to determine security requirements of users. Those should be left to the owners. If there is a security breach, management will ultimately be held responsible, so it should

make these
decisions in the first place.

Keep It Simple
The secure design principles we've discussed thus far can introduce complexity into our
information systems. As many of us have learned the hard way, the more complex a system is, the more difficult it is to understand and protect it. This is where the principle of
simplicity comes in: make everything as simple as possible and periodically check things
to ensure we are not adding unnecessary complexity.
This principle has long been known in the software development community, where one of the metrics routinely tracked is errors per 1,000 lines of code, also known as
defects per KLOC (kilo-lines of code). The idea is that the more code you write, the
greater the odds you'll make an error without noticing it. Similarly, the more unique
hosts, facilities, or policies you have, the greater the odds that a vulnerability will be
introduced inadvertently.
Obviously, we can't force a global enterprise to use a small network, but we can simplify things by standardizing configurations. We can have 10,000 endpoints around
the world, all configured in just a handful of ways with no exceptions allowed. We can
similarly ensure that all our facilities use the same security protocols and that our policies

PART III

As employees work at an organization over time and move from one department to another, they often are assigned more and more access rights and permissions. This
is commonly referred to as authorization creep. It can be a large risk for an organization, because too many users have too much privileged access to organizational
assets. In the past, it has usually been easier for network administrators to give more
access than less, because then the user would not come back and require more work
to be done on her profile. It is also difficult to know the exact access levels different
individuals require. This is why user management and user provisioning are becoming more prevalent in identity management products today and why organizations
are moving more toward role-based access control implementation. Enforcing least privilege on user accounts should be an ongoing job, which means each user's rights
are permissions that should be reviewed to ensure the organization is not putting
itself at risk.

are few in number, simple to understand, and uniformly enforced. The fewer variables we have to track, the simpler our systems become and, by extension, the easier they are to secure.

Secure Defaults

As many people in the technology field know, out-of-the-box implementations are oftentimes far from secure. Many systems come out of the box in an insecure state because settings have to be configured to properly integrate a system into different environments, and this is a friendlier way of installing the product for users. For example, if Mike is installing a new software package that continually throws messages of "Access Denied" when he is attempting to configure it to interoperate with other applications and systems, his patience might wear thin, and he might decide to hate that vendor for years to come because of the stress and confusion inflicted upon him.

Yet again, we are at a hard place for developers and architects. When a security application or device is installed, it should default to "No Access." This means that when Laurel installs a firewall, it should not allow any traffic to pass into the network that was not specifically granted access. A fine balance exists between security, functionality, and user friendliness. If an application is extremely user friendly or has lots of features, it is probably not as secure. It is up to us as security professionals to help our organizations balance these three competing needs.

NOTE Most operating systems nowadays ship with reasonably secure default settings, but users are still able to override the majority of these settings. This brings us closer to "default with no access," but we still have a ways to go.

The principle of secure defaults means that every system starts off in a state where security trumps user friendliness and functionality. It is later that, as deliberate decisions, security personnel can relax some restrictions, enable additional features, and generally make the system more user friendly. These decisions are integrated into the risk management program (see Chapter 2), typically through the configuration management processes (which we'll discuss in Chapter 20). The goal of secure defaults is to start everything in a place of extreme security and then intentionally loosen things until users can get their jobs done, but no further.

Fail Securely
A design principle that is related to secure defaults deals with the manner in which failures are handled. In the event of an error, information systems ought to be designed to behave in a predictable and noncompromising manner. This is also generally referred to as failing securely. We already saw how violating this principle can give adversaries an advantage when we discussed some of the cryptanalysis approaches in the previous chapter. If adversaries can induce a fault in a cryptosystem, they could recover a partial key or otherwise compromise it. The same holds for any information system.

A fail-secure system defaults to the highest level of security when it encounters a fault. For example, a firewall that is powered off unexpectedly may block all traffic when it reboots until an administrator can verify that it is still configured and operating securely. An EDR system may lock out a system if it encounters certain critical errors.
Finally, a web browser could prevent a user from visiting a website if the connection is not secure or the digital certificate doesn’t match, is not trusted, or is expired or revoked.
Many a successful phishing campaign could’ve been defeated by implementing this last example alone!

Privacy by Design

1. Proactive, not reactive; preventative, not remedial
2. Privacy as the default setting
3. Privacy embedded into design
4. Full functionality—positive-sum, not zero-sum
5. End-to-end security—full life-cycle protection
6. Visibility and transparency—keep it open
7. Respect for user privacy—keep it user-centric

Security Models
A security model is a more formal way to capture security principles. Whereas a principle is a rule of thumb that can be adapted to different situations, the security models we describe here are very specific and verifiable. A security model is usually represented in mathematics and analytical ideas, which are mapped to system specifications and then implemented in software and/or hardware by product developers. So we have a policy

that encompasses security goals, such as “each subject must be authenticated and authorized before accessing an object.” The security model takes this requirement and provides
the necessary mathematical formulas, relationships, and logic structure to be followed to
accomplish this goal. From there, specifications are developed per operating system type

PART III

The best way to ensure privacy of user data is to incorporate data protection as an
integral part of the design of an information system, not as an afterthought or laterstage feature. This is the principle of privacy by design in a nutshell. Apart from making sense (we hope), this principle is part of the European Union’s General Data
Protection Regulation (GDPR), which means your organization may be required to abide by it already.
Privacy by design is not a new concept. It was originally introduced in the 1990s and
formally described in a joint report by the Dutch Data Protection Authority and the
Ontario Information Commissioner, titled Privacy by Design: Delivering the Promises, in
2010. This document describes the seven foundational principles of privacy by design,
which are listed here:

(Unix, Windows, macOS, and so on), and individual vendors can decide how they are
going to implement mechanisms that meet these necessary specifications.
Several security models have been developed to enforce security policies. The following sections provide overviews of the models with which you must be familiar as
a CISSP.

Bell-LaPadula Model
The Bell-LaPadula model enforces the confidentiality aspects of access control. It was
developed in the 1970s to prevent secret information from being accessed in an unauthorized manner. It was the first mathematical model of a multilevel security policy
used to define the concept of secure modes of access and outlined rules of access. Its
development was funded by the U.S. government to provide a framework for computer systems that would be used to store and process sensitive information. A system
that employs the Bell-LaPadula model is called a multilevel security system because users
with different clearances use the system, and the system processes data at different classification levels.

EXAM TIP The Bell-LaPadula model was developed to make sure secrets stay secret; thus, it provides and addresses confidentiality only. This model does not address the integrity of the data the system maintains—only who can and cannot access the data and what operations can be carried out.

Three main rules are used and enforced in the Bell-LaPadula model:

- Simple security rule
- *-property (star property) rule
- Strong star property rule

The simple security rule states that a subject at a given security level cannot read data that resides at a higher security level. For example, if Bob is given the security clearance of secret, this rule states he cannot read data classified as top secret. If the organization wanted Bob to be able to read top-secret data, it would have given him that clearance in the first place.

The *-property rule (star property rule) states that a subject in a given security level cannot write information to a lower security level. The simple security rule is referred to as the "no read up" rule, and the *-property rule is referred to as the "no write down" rule.

The strong star property rule states that a subject who has read and write capabilities can only perform both of those functions at the same security level; nothing higher and nothing lower. So, for a subject to be able to read and write to an object, the subject's clearance and the object classification must be equal.

Biba Model

The Biba model is a security model that addresses the integrity of data within a system. It is not concerned with security levels and confidentiality. The Biba model uses integrity levels to prevent data at any integrity level from flowing to a higher integrity level. Biba has three main rules to provide this type of protection:

- *-integrity axiom A subject cannot write data to an object at a higher integrity level (referred to as "no write up").
- Simple integrity axiom A subject cannot read data from a lower integrity level (referred to as "no read down").
- Invocation property A subject cannot request service (invoke) at a higher

integrity.

Bell-LaPadula vs. Biba
The Bell-LaPadula and Biba models are informational flow models because they are most concerned about data flowing from one level to another. Bell-LaPadula uses security levels to provide data confidentiality, and Biba uses integrity levels to
provide data integrity.
It is important for CISSP test takers to know the rules of Bell-LaPadula and Biba,
and their rules sound similar. Both have “simple” and “* (star)” rules—one writing
one way and one reading another way. A tip for how to remember them is if the word
“simple” is used, the rule is about reading. If the rule uses * or “star,” it is about writing.
So now you just need to remember the reading and writing directions per model.

PART III

A simple example might help illustrate how the Biba model could be used in a real
context. Suppose that Indira and Erik are on a project team and are writing two documents:
Indira is drafting meeting notes for internal use and Erik is writing a report for the CEO.
The information Erik uses in writing his report must be very accurate and reliable, which
is to say it must have a high level of integrity. Indira, on the other hand, is just documenting
the internal work being done by the team, including ideas, opinions, and hunches. She
could use unconfirmed and maybe even unreliable sources when writing her document.
The *-integrity axiom dictates that Indira would not be able to contribute (write) material
to Erik’s report, though there’s nothing to say she couldn’t use Erik’s (higher integrity)
information in her own document. The simple integrity axiom, on the other hand, would
prevent Erik from even reading Indira’s document because it could potentially introduce
lower integrity information into his own (high integrity) report.
The invocation property in the Biba model states that a subject cannot invoke (call
upon) a subject at a higher integrity level. How is this different from the other two
Biba rules? The *-integrity axiom (no write up) dictates how subjects can modify objects.
The simple integrity axiom (no read down) dictates how subjects can read objects. The
invocation property dictates how one subject can communicate with and initialize other

subjects at run time. An example of a subject invoking another subject is when a process sends a request to a procedure to carry out some type of task. Subjects are only allowed to invoke tools at a lower integrity level. With the invocation property, the system is making sure a dirty subject cannot invoke a clean tool to contaminate a clean object.

Clark-Wilson Model

The Clark-Wilson model was developed after Biba and takes some different approaches to protecting the integrity of information. This model uses the following elements:

- Users Active agents
- Transformation procedures (TPs) Programmed abstract operations, such as read, write, and modify
- Constrained data items (CDIs) Can be manipulated only by TPs
- Unconstrained data items (UDIs) Can be manipulated by users via primitive read and write operations
- Integrity verification procedures (IVPs) Check the consistency of CDIs with external reality

A distinctive feature of the Clark-Wilson model is that it focuses on well-formed transactions and separation of duties. A well-formed transaction is a series of operations that transforms a data item from one consistent state to another. Think of a consistent state as one wherein we know the data is reliable. This consistency ensures the integrity of the data and is the job of the TPs. Separation of duties is implemented in the model by adding a type of procedure (the IVPs) that audits the work done by the TPs and validates the integrity of the data.

When a system uses the Clark-Wilson model, it separates data into one subset that needs to be highly protected, which is referred to as a constrained data item (CDI), and another subset that does not require a high level of protection, which is called an unconstrained data item (UDI). Users cannot modify critical data (CDI) directly. Instead, software procedures (TPs) carry out the operations on behalf of the users. This is referred to as access triple: subject (user), program (TP), and object (CDI). A user cannot modify a CDI without using a TP. The UDI does not require such a high level of protection and can be manipulated directly by the user.

Remember that this is an integrity model, so it must have something that ensures that

specific integrity rules are being carried out. This is the job of the IVP. The IVP ensures
that all critical data (CDI) manipulation follows the application’s defined integrity rules.

Noninterference Model
Multilevel security properties can be expressed in many ways, one being noninterference.
This concept is implemented to ensure any actions that take place at a higher security
level do not affect, or interfere with, actions that take place at a lower level. This type
of model does not concern itself with the flow of data, but rather with what a subject

Covert Channels
A covert channel is a way for an entity to receive information in an unauthorized
manner. These communications can be very difficult to detect. Covert channels
are of two types: storage and timing. In a covert storage channel, processes are able
to communicate through some type of storage space on the system. For example,
suppose Adam wants to leak classified information to Bob. Adam could create a
user account on a web system. Bob pretends he will create an account on the same
system and checks to see if the username is available. If it is available, that is the
equivalent of a zero (no account existed). Otherwise, he records a one, and aborts
the creation of the account. Either way, Bob waits a given amount of time. Adam
either removes the account, effectively writing a zero, or ensures one exists (which
would be a one). Bob tries again, recording the next bit of covertly communicated
information.
In a covert timing channel, one process relays information to another by
modulating its use of system resources. Adam could tie up a shared resource (such
as a communications bus). Bob tries to access the resource and, if successful, records
it as a zero bit (no wait). Otherwise, he records a one and waits a predetermined
amount of time. Adam, meanwhile, is encoding his covert message by selectively
tying up or freeing the shared resource. Think of this as a type of Morse code, but
using some type of system resource.

knows about the state of the system. So, if an entity at a higher security level performs an

action, it cannot change the state for the entity at the lower level. If a lower-level entity was aware of a certain activity that took place by an entity at a higher level and the state of the system changed for this lower-level entity, the entity might be able to deduce too much information about the activities of the higher state, which, in turn, is a way of leaking information.

Let's say that Tom and Kathy are both working on a multilevel mainframe at the same time. Tom has the security clearance of secret, and Kathy has the security clearance of top secret. Since this is a central mainframe, the terminal Tom is working at has the context of secret, and Kathy is working at her own terminal, which has a context of top secret.

This model states that nothing Kathy does at her terminal should directly or indirectly affect Tom's domain (available resources and working environment). The commands she executes or the resources she interacts with should not affect Tom's experience of working with the mainframe in any way.

The real intent of the noninterference model is to address covert channels. The model looks at the shared resources that the different users of a system will access and tries to identify how information can be passed from a process working at a higher security clearance to a process working at a lower security clearance. Since Tom and Kathy are working on the same system at the same time, they will most likely have to share some types of resources. So the model is made up of rules to ensure that Kathy cannot pass data to Tom through covert channels.

Brewer and Nash Model

The Brewer and Nash model, also called the Chinese Wall model, states that a subject can write to an object if, and only if, the subject cannot read another object that is in a different dataset. It was created to provide access controls that can change dynamically depending upon a user's previous actions. The main goal of the model is to protect against conflicts of interest by users' access attempts. Suppose Maria is a broker at an investment firm that also provides other services to Acme Corporation. If Maria were able

to access
Acme information from the other service areas, she could learn of a phenomenal earnings
report that is about to be released. Armed with that information, she could encourage her
clients to buy shares of Acme, confident that the price will go up shortly. The Brewer and
Nash Model is designed to mitigate the risk of this situation happening.

Graham-Denning Model
Remember that these are all models, so they are not very specific in nature. Each individual
vendor must decide how it is going to actually meet the rules outlined in the chosen model.
Bell-LaPadula and Biba do not define how the security and integrity levels are defined and
modified, nor do they provide a way to delegate or transfer access rights. The GrahamDenning model addresses some of these issues and defines a set of basic rights in terms of
commands that a specific subject can execute on an object. This model has eight primitive
protection rights, or rules of how these types of functionalities should take place securely:

- How to securely create an object
- How to securely create a subject
- How to securely delete an object
- How to securely delete a subject
- How to securely provide the read access right
- How to securely provide the grant access right
- How to securely provide the delete access right
- How to securely provide transfer access rights

These functionalities may sound insignificant, but when you’re building a secure system,
they are critical. If a software developer does not integrate these functionalities in a secure
manner, they can be compromised by an attacker and the whole system can be at risk.

Harrison-Ruzzo-Ullman Model
The Harrison-Ruzzo-Ullman (HRU) model deals with access rights of subjects and the
integrity of those rights. A subject can carry out only a finite set of operations on an
object. Since security loves simplicity, it is easier for a system to allow or disallow authorization of operations if one command is restricted to a single operation. For example,
if a subject sent command X that only requires the operation of Y, this is pretty straightforward and the system can allow or disallow this operation to take place. But if a subject

Security Models Recap
All of these models can seem confusing. Most people are not familiar with all of them, which can make the information even harder to absorb. The following are the
core concepts of the different models.
Bell-LaPadula Model This is the first mathematical model of a multilevel security
policy that defines the concept of a secure state and necessary modes of access. It ensures that information only flows in a manner that does not violate the system
policy and is confidentiality focused.

Biba Model A model that describes a set of access control rules designed to ensure
data integrity.

• The simple integrity axiom A subject cannot read data at a lower integrity level (no read down).
• The *-integrity axiom A subject cannot modify an object at a higher integrity level (no write up).
Clark-Wilson Model This integrity model is implemented to protect the integrity of data and to ensure that properly formatted transactions take place. It addresses
all three goals of integrity:

• Subjects can access objects only through authorized programs (access triple).
• Separation of duties is enforced.
• Auditing is required.
Noninterference Model This formal multilevel security model states that commands and activities performed at one security level should not be seen by, or
affect, subjects or objects at a different security level.
Brewer and Nash Model This model allows for dynamically changing access controls that protect against conflicts of interest. Also known as the Chinese Wall model.
Graham-Denning Model This model shows how subjects and objects should be created and deleted. It also addresses how to assign specific access rights.
Harrison-Ruzzo-Ullman Model This model shows how a finite set of procedures can be available to edit the access rights of a subject.

PART III

• The simple security rule A subject cannot read data within an object that resides at a higher security level (no read up).
• The *-property rule A subject cannot write to an object at a lower security level (no write down).
• The strong star property rule For a subject to be able to read and write to an object, the subject’s clearance and the object’s classification must be equal.

sent a command M and to fulfill that command, operations N, B, W, and P have to

be
carried out, then there is much more complexity for the system to decide if this command should be authorized.
Also the integrity of the access rights needs to be ensured; thus, in this example, if one
operation cannot be processed properly, the whole command fails. So although it is easy
to dictate that subject A can only read object B, it is not always so easy to ensure each and
every function supports this high-level statement. The HRU model is used by software
designers to ensure that no unforeseen vulnerability is introduced and the stated access
control goals are achieved.

Security Requirements

Whether we are building enterprise security architectures or software systems or anything
in between, we always want to start from a set of requirements. These are what ultimately
tell us whether or not the thing we built meets our needs. Security requirements should
flow out of the organizational risk management processes, be informed by threat models,
and be grounded in the principles and models we discussed earlier in this chapter. These
requirements are then addressed within the frameworks we discussed in Chapter 4, and
the whole thing is then assessed over time using a maturity model like the Capability
Maturity Model Integration (CMMI), also discussed in Chapter 4.
One of the key tasks in addressing security requirements in our enterprise architectures
is selecting the right controls for each requirement and then implementing, documenting,
and verifying them. The exact process will vary depending on the framework you choose,
but you may want to review the CRM example we discussed in Chapter 4 that applies
NIST SP 800-53 (see Table 4-2 in particular).

Security Capabilities of Information Systems

Satisfying security requirements has become easier over the years as most vendors now
incorporate advanced security capabilities into their products, particularly physical ones.
After all, we can go to great lengths to ensure that the software we develop is secure, to
run it on operating systems that have been hardened by a myriad of security controls,
and to monitor everything using advanced security tools, but if the physical devices on
which these systems run are untrustworthy, then all our efforts are for naught.
In the sections that follow, we discuss a variety of hardware-based capabilities

of many information
systems that you should know.

Trusted Platform Module
A Trusted Platform Module (TPM) is a hardware component installed on the motherboard
of modern computers that is dedicated to carrying out security functions involving the
storage of cryptographic keys and digital certificates, symmetric and asymmetric
encryption, and hashing. The TPM was devised by the Trusted Computing Group (TCG),
an organization that promotes open standards to help strengthen computing platforms
against security weaknesses and attacks.

The essence of a TPM lies in a protected and encapsulated microcontroller security
chip that provides a safe haven for storing and processing critical security data such as
keys, passwords, and digital certificates. The use of a dedicated and encoded
hardwarebased platform drastically improves the root of trust of the computing system, while
allowing for a vastly superior implementation and integration of security features. The
introduction of TPM has made it much harder to access information on computing
devices without proper authorization and allows for effective detection of malicious
configuration changes to a computing platform.

TPM Uses

Persistent Memory Two kinds of keys are present in the static memory:

• Endorsement Key (EK) A public/private key pair that is installed in the TPM at the time of manufacture and cannot be modified. The private key is always present
inside the TPM, while the public key is used to verify the authenticity of the TPM
itself. The EK, installed in the TPM, is unique to that TPM and its platform.
• Storage Root Key (SRK) The master wrapping key used to secure the keys stored in the TPM.

PART III

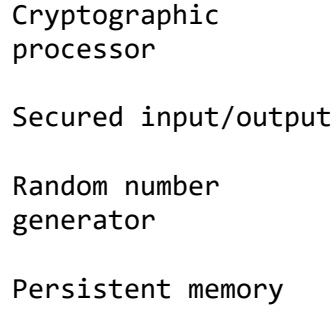
The most common usage scenario of a TPM is to bind a hard disk drive, where the content of a given hard disk drive is affixed with a particular computing system. The content
of the hard disk drive is encrypted, and the decryption key is stored away in a TPM
chip. To ensure safe storage of the decryption key, it is further “wrapped” with another

encryption key. Binding a hard disk drive makes its content basically inaccessible to other systems, and any attempt to retrieve the drive's content by attaching it to another system will be very difficult. However, in the event of a TPM chip's failure, the hard drive's content will be rendered useless, unless a backup of the key has been escrowed.

Another application of a TPM is sealing a system's state to a particular hardware and software configuration. Sealing a computing system through TPM is used to deter any attempts to tamper with a system's configurations. In practice, this is similar to how hashes are used to verify the integrity of files shared over the Internet (or any other untrusted medium). Sealing a system is fairly straightforward. The TPM generates hash values based on the system's configuration files and stores them in its memory. A sealed system will be activated only after the TPM verifies the integrity of the system's configuration by comparing it with the original "sealing" value.

A TPM is essentially a securely designed microcontroller with added modules to perform cryptographic functions. These modules allow for accelerated storage and processing of cryptographic keys, hash values, and pseudonumber sequences. A TPM's internal storage is based on random access memory (RAM), which retains its information when power is turned off and is therefore termed nonvolatile RAM (NVRAM). A TPM's internal memory is divided into two different segments: persistent (static) and versatile (dynamic) memory modules, as shown in Figure 9-4.

Figure 9-4 Functional components of a Trusted Platform Module



Storage Root
Key (SRK)
RSA key generator
Versatile memory
Platform Configuration
Registers (PCR)
SHA-1 hash generator
Attestation Identity
Key (AIK)
Encryption-decryptionsignature engine

Storage keys

Versatile Memory Three kinds of keys (or values) are present in the versatile memory:

• Platform Configuration Registers (PCRs) Used to store cryptographic hashes of data used for TPM's sealing functionality.
• Attestation Identity Keys (AIKs) Used for the attestation of the TPM chip itself to service providers. The AIK is linked to the TPM's identity at the time of development, which in turn is linked to the TPM's EK. Therefore, the AIK ensures the integrity of the EK.
• Storage keys Used to encrypt the storage media of the computer system.

Hardware Security Module
Whereas a TPM is a microchip installed on a motherboard, a hardware security module
(HSM) is a removable expansion card or external device that can generate, store, and manage cryptographic keys. HSMs are commonly used to improve encryption/decryption
performance by offloading these functions to a specialized module, thus freeing up the
general-purpose microprocessor to take care of, well, general-purpose tasks. HSMs have
become critical components for data confidentiality and integrity in digital business
transactions. The U.S. Federal Information Processing Standard (FIPS) 140-2 is perhaps
the most widely recognized standard for evaluating the security of an HSM. This evaluation is important, because so much digital commerce nowadays relies on protections
provided by HSM.

As with so many other cybersecurity technologies, the line between TPMs and HSMs gets blurred. TPMs are typically soldered onto a motherboard, but they can be added
through a header. HSMs are almost always external devices, but you will occasionally see
them as Peripheral Component Interconnect (PCI) cards. In general, however, TPMs are permanently mounted and used for hardware-based assurance and key storage, while

HSMs are removable (or altogether external) and are used for both hardware-accelerated cryptography and key storage.

Self-Encrypting Drive

NOTE

Although SEDs can use onboard TPMs, that is not the norm.

The data stored in an SED is encrypted using symmetric key encryption, and it's decrypted dynamically whenever the device is read. A write operation works the other way—that is, the plaintext data arrives at the drive and is encrypted automatically before being stored on disk. Because the SED has its own hardware-based encryption engine, it tends to be faster than software-based approaches.
Encryption typically uses the Advanced Encryption Standard (AES) and a 128- or 256-bit key. This secret key is stored in nonvolatile memory within the cryptographic module and is itself encrypted with a password chosen by the user. If the user changes the password, the same secret key is encrypted with the new password, which means the whole disk doesn't have to be decrypted and then re-encrypted. If ever there is a need to securely wipe the contents of the SED, the cryptographic module is simply told to generate a new secret key. Since the drive contents were encrypted with the previous (now overwritten) key, that data is effectively wiped. As you can imagine, wiping an SED is almost instantaneous.

Bus Encryption
While the self-encrypting drive protects the data as it rests on the drive, it decrypts the data prior to transferring it to memory for use. This means that an attacker has three opportunities to access the plaintext data: on the external bus connecting the drive to the motherboard (which is sometimes an external cable), in memory, or on the bus between memory and the CPU. What if we moved the cryptographic module from the disk controller to the CPU? This would make it impossible for attackers to access the plaintext data outside the CPU itself, making their job that much harder.

PART III

Full-disk encryption (FDE) refers to approaches used to encrypt the entirety of data at rest on a disk drive. This can be accomplished in either software or hardware. A

self-encrypting
drive (SED) is a hardware-based approach to FDE in which a cryptographic module is
integrated with the storage media into one package. Typically, this module is built right
into the disk controller chip. Most SEDs are built in accordance with the TCG Opal 2.0
standard specification.

Bus encryption means data and instructions are encrypted prior to being put on the
internal bus, which means they are also encrypted everywhere else except when data
is being processed. This approach requires a specialized chip, a cryptoprocessor, that
combines traditional CPU features with a cryptographic module and specially protected
memory for keys. If that sounds a lot like a TPM, it’s because it usually is.
You won’t see bus encryption in general-purpose computers, mostly because the
cryptoprocessors are both more expensive and less capable (performance-wise) than
regular CPUs. However, bus encryption is a common approach to protecting highly
sensitive systems such as automated teller machines (ATMs), satellite television boxes,
and military weapon systems. Bus encryption is also widely used for smart cards. All
these examples are specialized systems that don’t require a lot of processing power but do
require a lot of protection from any attacker who gets his or her hands on them.

Secure Processing
By way of review, data can exist in one of three states: at rest, in transit, or in use. While
we’ve seen how encryption can help us protect data in the first two states, it becomes a bit
trickier when it is in use. The reason is that processors almost always need unencrypted
code and data to work on.
There are three common ways to protect data while it’s in use. The first is to create a
specially protected part of the computer in which only trusted applications can run with
little or no interaction with each other or those outside the trusted environment. Another
approach is to build extensions into the processors that enable them to create miniature
protected environments for each application (instead of putting them all together in
one trusted environment). Finally, we can just write applications that temporarily lock
the processor and/or other resources to ensure nobody interferes with them until

they’re
done with a specific task. Let’s take a look at these approaches in order.

Trusted Execution Environment
A trusted execution environment (TEE) is a software environment in which special applications and resources (such as files) have undergone rigorous checks to ensure that they are
trustworthy and remain protected. Some TEEs, particularly those used in Apple products, are called secure enclaves, but the two terms are otherwise interchangeable. TEEs
exist in parallel with untrusted rich execution environments (REEs) on the same platform,
as shown in Figure 9-5. TEEs are widely used in mobile devices and increasingly included
in embedded and IoT devices as well, to ensure that certain critical applications and their
data have guaranteed confidentiality, integrity, and availability. TEEs are also starting
to show up in other places, such as microservices and cloud services, where hardware
resources are widely shared.
A TEE works by creating a trust boundary around itself and strictly controlling the way in which the untrusted REE interacts with the trusted applications. The TEE
typically has its own hardware resources (such as a processor core, memory, and persistent
storage) that are unavailable to the REE. It also runs its own trusted OS that is separate
from and independent of the one in the REE. The two environments interact through a

App

Trusted execution
environment

Trusted
app

App

OS

Bootloader

Trusted
app

Trusted OS

Trusted bootloader

Trust boundary

Rich execution
environment

TEE external API

Figure 9-5
A typical TEE and
its related REE

restricted external application programming interface (API) that enables the rich OS to
call a limited set of services provided by the REE.
NOTE The term “secure enclave” is most commonly associated with Apple products such as the iPhone, but it is otherwise equivalent to the term “trusted execution environment.”

So, how do TPMs, HSMs, and TEEs differ from each other? A TPM is usually a system on a chip (SoC) soldered onto the motherboard to provide limited cryptographic
functions. An HSM is a big TPM that plugs into a computer system to provide these
functions at a much larger scale. A TEE can perform the functions of a TPM, but, unlike
both the TPM and HSM, it is specifically designed to run trusted applications that may
have nothing to do with cryptography.
Trusted execution starts with a secure boot, in which the firmware verifies the integrity
of the trusted OS bootloader before executing it. In fact, every executable and driver
in the TEE is verified to the hardware root of trust and restricted to its own assigned
resources. Only specific applications that have undergone rigorous security assessments
at the hands of trusted parties are deployed in the TEE by the device manufacturer.
This enables trusted applications such as cryptography, identity, and payment systems to
enjoy high levels of protection that would otherwise be impossible to attain.
EXAM TIP TEEs (and, by extension, secure enclaves) do not implement hardware roots of trust because they are implemented in software. However, TEEs typically rely on an underlying root of trust provided by a TPM on the device.

PART III

Firmware

Processor Security Extensions

TEEs need hardware support, which all the major chip manufacturers provide in their chipsets. Security is baked into the chips of most modern microprocessors. These CPU packages become a security perimeter outside of which all data and code can exist in encrypted form. Before encrypted data or code can cross into the secure perimeter, it can be decrypted and/or checked for integrity. Even once allowed inside, data and code are restricted by special controls that ensure what may be done with or to them. For all this to work, however, we need to enable the features through special instructions.

Processor security extensions are instructions that provide these security features in the CPU and can be used to support a TEE. They can, for example, enable programmers to designate special regions in memory as being encrypted and private for a given process.

These regions are dynamically decrypted by the CPU while in use, which means any unauthorized process, including the OS or a hypervisor, is unable to access the plaintext stored in them. This feature is one of the building blocks of TEEs, which enables trusted applications to have their own protected memory.

Atomic Execution

Atomic execution is an approach to controlling the manner in which certain sections of a program run so that they cannot be interrupted between the start and end of a section.

This prevents other processes from interfering with resources being used by the protected process. To enable this, the programmer designates a section of code as atomic by placing a lock around it. The compiler then leverages OS libraries that, in turn, invoke hardware protections during execution of that locked code segment. The catch is that if you do this too often, you will see some dramatic performance degradation in a modern multithreaded OS. You want to use atomic execution as little as possible to protect critical resources and tasks.

Atomic execution protects against a class of attacks called time-of-check to time-ofuse (TOC/TOU). This type of attack exploits the dependency on the timing of events that take place in a multitasking OS. When running a program, an OS must carry out instruction 1, then instruction 2, then instruction 3, and so on. This is how programs are normally written. If an attacker can get in between instructions 2 and 3 and manipulate something, she can control the result of these activities. Suppose instruction 1

verifies
that a user is authorized to read an unimportant file that is passed as a link, say, a help
file. Instruction 2 then opens the file pointed to by the link, and instruction 3 closes
it after it’s been read by the user. If an attacker can interrupt this flow of execution
after instruction 1, change the link to point to a sensitive document, and then allow
instruction 2 to execute, the attacker will be able to read the sensitive file even though she
isn’t authorized to do so. By enforcing atomic execution of instructions 1 and 2 together,
we would protect against TOC/TOU attacks.
NOTE This type of attack is also referred to as an asynchronous attack.
Asynchronous describes a process in which the timing of each step may vary.
The attacker gets in between these steps and modifies something.

411
Putting It All Together: Where Can Data Be Encrypted?

Disk

Disk
Crypto

Disk

Disk

Memory

Memory

External bus

Memory

Memory

Internal
bus

CPU

CPU

Standard
system

Selfencrypting
drive

Crypto
Processor
Bus
encryption

REE

Crypto

TEE

Trusted
execution
environment

Chapter Review
One of the keys to providing the best security possible is to have a baseline understanding of adversaries that may target the organization, what their capabilities are, and what
motivates them. We saw multiple ways to do that in this chapter, of which the MITRE
ATT&CK framework is probably the one you want to dig into on your own. It seems like many professionals and organizations alike are converging on it as the lingua franca
by which to describe adversarial behaviors.
A sound approach to defeating these threat actors is to apply the fundamental principles
of secure design, of which we covered the 11 that ISC2 stresses in the CISSP Certification

PART III

Data in a computer system can exist in three different places: in the processor, in memory, and in secondary storage such as a disk drive. Standard systems do not encrypt data in any of these three places by default. You can opt to use FED,
such as a self-encrypting drive, to encrypt the data in secondary storage, but that
leaves it exposed everywhere else in the system, including the external bus. The third
option is to use bus encryption, which requires a cryptoprocessor that is (relatively)
expensive and underpowered. You are unlikely to want this unless you really have to
protect the data in situations where you assume the adversary will be able to hack
your hardware. Finally, the most flexible (and common) balance of protection, performance, and cost is the use of TEEs that can coexist with untrusted applications.
Only the data within the TEE receives the full encryption treatment outside the CPU, leaving everything else to run on regular processor cores.

Exam Outline. There are other principles that you may be tracking, but these are the 11 you'll need to know for the exam. Likewise, the security models we discussed, which bring extra rigor to the study of security, are sure to make an appearance in the exam.
Pay particular attention to Biba and Bell-LaPadula. Together, these principles and models provide a solid foundation on which to select controls based upon systems security requirements and build a solid security architecture.

Quick Review
• Threat modeling is the process of describing probable adverse effects on our assets caused by specific threat sources.
• An attack tree is a graph showing how individual actions by attackers can be chained together to achieve their goals.
• STRIDE is a threat modeling framework developed by Microsoft that evaluates a system's design using flow diagrams, system entities, and events related to a system.
• The Lockheed Martin Cyber Kill Chain identifies seven stages of cyberattacks.
• The MITRE ATT&CK framework is a comprehensive matrix of tactics and techniques used to model cyberattacks.
• Defense in depth is the coordinated use of multiple security controls in a layered approach.
• Zero trust is a model in which every entity is considered hostile until proven otherwise, and even that trust is limited.
• Trust but verify is the principle that, even when an entity and its behaviors are trusted, we should double-check both.
• Shared responsibility refers to the situation in which a service provider is responsible for certain security controls, while the customer is responsible for others.
• Separation of duties divides important functions among multiple individuals to ensure that no one person has the ability to intentionally or accidentally cause serious losses to the organization.
• Least privilege states that people are granted exactly the access and authority that they require to do their jobs, and nothing more.
• The need-to-know principle, which is similar to the least-privilege principle, is based on the concept that individuals should be given access only to the information they absolutely require in order to perform their job duties.
• The "keep it simple" principle drives us to make everything as simple as possible and periodically check things to ensure we are not adding unnecessary complexity.
• The principle of secure defaults means that every system starts off in a state where security trumps user friendliness and functionality.

Questions
Please remember that these questions are formatted and asked in a certain way for a
reason. Keep in mind that the CISSP exam is asking questions at a conceptual level.
Questions may not always have the perfect answer, and the candidate is advised against
always looking for the perfect answer. Instead, the candidate should look for the best
answer in the list.

PART III

• The principle of failing securely states that, in the event of an error, information
systems ought to be designed to behave in a predictable and noncompromising manner.
• The principle of privacy by design states that the best way to ensure privacy of user data is to incorporate data protection as an integral part of the design of
an information system, not as an afterthought or later-stage feature.
• The Bell-LaPadula model enforces the confidentiality aspects of access control.
• The Biba model is a security model that addresses the integrity of data within a system but is not concerned with security levels and confidentiality.
• The Brewer and Nash model, also called the Chinese Wall model, states that a subject can write to an object if, and only if, the subject cannot read another
object that is in a different dataset.
• A Trusted Platform Module (TPM) is dedicated to carrying out security functions
involving the storage of cryptographic keys and digital certificates, symmetric and
asymmetric encryption, and hashing.
• A hardware security module (HSM) is a removable expansion card or external device that can generate, store, and manage cryptographic keys to improve encryption/decryption performance of the system into which it is installed.
• A self-encrypting drive (SED) provides full disk encryption (FDE) through a cryptographic module that is integrated with the storage media into one package.
• Data in SEDs is encrypted using symmetric key cryptography.
• Bus encryption systems use TPMs to encrypt data and instructions prior to being
put on the internal bus, which means they are also encrypted everywhere else except when data is being processed.
• A trusted execution environment (TEE), or a secure enclave, is a software environment in which special applications and resources (such as files) have undergone rigorous checks to ensure that they are trustworthy and remain protected.
• Processor security extensions are instructions that provide additional

security
features in the CPU and can be used to support a TEE.
• Atomic execution is an approach to controlling the manner in which certain sections of a program run so that they cannot be interrupted between the start and end of the section.

1. Developed by Microsoft, which threat-modeling technique is suitable for application to logical and physical systems alike?
A. Attack trees
B. STRIDE
C. The MITRE ATT&CK framework
D. The Cyber Kill Chain

2. Which threat modeling framework provides detailed procedures followed by specific cyberthreat actors?
A. Attack trees
B. STRIDE
C. The MITRE ATT&CK framework
D. The Cyber Kill Chain

3. Which of the following security models is concerned with the confidentiality and
not the integrity of information?
A. Biba
B. Bell-LaPadula
C. Brewer and Nash
D. Clark-Wilson

4. Which of the following security models is concerned with the integrity and not
the confidentiality of information?
A. Biba
B. Bell-LaPadula
C. Graham-Denning
D. Brewer and Nash

5. Where is the data encrypted in a self-encrypting drive system?
A. On the disk drive
B. In memory
C. On the bus
D. All of the above

6. Where is the data encrypted in a bus encryption system?
A. On the disk drive
B. In memory
C. On the bus
D. All of the above

7. What is the difference between a Trusted Platform Module (TPM) and a hardware security module (HSM)?
A. An HSM is typically on the motherboard and a TPM is an external device.
B. Only an HSM can store multiple digital certificates.
C. There is no difference, as both terms refer to the same type of device.
D. A TPM is typically on the motherboard and an HSM is an external device.

8. Which of the following is not a required feature in a TPM?
A. Hashing
B. Certificate revocation
C. Certificate storage

9. Which of the following is true about changing the password on a self-encrypting
drive?
A. It requires re-encryption of stored data.
B. The new password is encrypted with the existing secret key.
C. It has no effect on the encrypted data.
D. It causes a new secret key to be generated.

10. Which of these is true about processor security extensions?
A. They are after-market additions by third parties.
B. They must be disabled to establish trusted execution environments.
C. They enable developers to encrypt memory associated with a process.
D. Encryption is not normally one of their features.

Answers
1. B. STRIDE is a threat-modeling framework that evaluates a system’s design using
flow diagrams, system entities, and events related to a system.
2. C. The MITRE ATT&CK framework maps cyberthreat actor tactics to the techniques used for them and the detailed procedures used by specific threat actors during cyberattacks.
3. B. The Bell-LaPadula model enforces the confidentiality aspects of access control.
4. A. The Biba model is a security model that addresses the integrity of data within
a system but is not concerned with security levels and confidentiality.
5. A. Self-encrypting drives include a hardware module that decrypts the data prior to putting it on the external bus, so the data is protected only on the drive itself.

PART III

D. Encryption

6. D. In systems that incorporate bus encryption, the data is decrypted only on the cryptoprocessor. This means that the data is encrypted everywhere else on the system.
7. D. In general, TPMs are permanently mounted on the motherboard and used for hardware-based assurance and key storage, while HSMs are removable or

altogether external and are used for both hardware accelerated cryptography and key storage.
8. B. Certificate revocation is not a required feature in a TPM. TPMs must provide storage of cryptographic keys and digital certificates, symmetric and asymmetric encryption, and hashing.
9. C. When you change the password on a self-encrypting drive, the existing secret key is retained but is encrypted with the new password. This means the encrypted data on the disk remains unaltered.
10. C. Processor security extensions are instructions that provide security features in the CPU and can be used to support a trusted execution environment. They can, for example, enable programmers to designate special regions in memory as being encrypted and private for a given process.

CHAPTER

Site and Facility Security

This chapter presents the following:
- Security principles of facility design
- Designing facility security controls

A building has at least two lives—the one imagined by its maker and the life it lives afterward—and they are never the same.
—Rem Koolhaas

We close out the third domain of the CISSP Common Body of Knowledge (CBK) by turning our attention to a topic to which many of us cybersecurity professionals don't pay enough attention: the security of our facilities and buildings. Most of us are focused on people and technology, but without a secure physical environment, all these efforts could be for naught. If adversaries can put their hands on our computers at will, it becomes much more difficult to keep them from also getting their hands on our information.

In this chapter, we take a good look at all that goes into securing the facilities that house our people, equipment, and information. Whether you get to build a site from scratch, have to choose an existing one, or are already occupying one, you should know and be able to apply the security principles we'll discuss here. We'll start off with the planning and design processes. Next, we'll examine how to apply the secure design principles (discussed in the previous chapter) to the overall design of a site or facility.

We'll then explore how to refine that design by selecting specific controls that mitigate risks to tolerable levels. Although we don't explicitly cover it in this chapter

(and just as
with any other aspect of security), we must periodically review and test our plans and
controls so that they remain effective and are continuously improved.

Site and Facility Design
The terms site and facility are oftentimes used interchangeably, and although the CISSP
exam does not make a strong distinction between them, we should clarify what they each
mean for purposes of this discussion. A site is a geographic area with fixed boundaries
that typically contains at least one building and its supporting structures (e.g., a parking
lot or electric substation). A facility is a building or a part of a building dedicated to a
specific purpose, such as corporate headquarters or a data center. So, a site would include

one or more facilities within it. Sometimes, an organization will have a facility inside
someone else’s site or even building, such as when an organization rents a group of connected offices (the facility) in a corporate plaza (the site).
EXAM TIP Don’t worry about differentiating the terms site and facility for purposes of the exam.

Site planning, like almost anything else, starts with a good set of requirements. These
depend upon the level of protection required for the various assets and the organization as
a whole. This required level of protection, in turn, is determined by the risk management
processes we discussed in Chapter 2, particularly the risk assessment. Physical security is a
combination of structures, people, processes, procedures, technology, and equipment to
protect resources. The design of a solid physical security program should be methodical
and should weigh the objectives of the program and the available resources. Although
every organization is different, the approach to constructing and maintaining a physical
security program is the same. The organization must first define the vulnerabilities,
threats, threat agents, and targets, which may be different than the ones we normally
track in cybersecurity.
NOTE Remember that a vulnerability is a weakness and a threat is the
event or mechanism that could actually exploit this identified vulnerability.

The threat agent is the person or thing that initiates the threat against this identified vulnerability.

Security Principles

Let’s take a moment to review the security principles covered in Chapter 9, which are equally applicable to designing secure networks and designing secure facilities. In the sections that follow, we briefly point out some examples of how these principles are applied in real organizations. We could provide many more examples, but the point is to show how the principles apply, not to be all-inclusive.

EXAM TIP You should be prepared to identify the application of the principles of secure design in a given scenario on the exam.

Threat Modeling

Securing anything, physical facilities included, should start with the question: securing it from what? Depending on the nature of our organizations and their environments, our concerns may range from petty thieves to terrorists. If we were to hold a brainstorming session, we could probably think of a very large set of potential threat actors carrying out an even larger set of harmful actions. It is helpful to narrow things down a bit by considering the most likely threat and then the most dangerous one too. For example,

Defense in Depth

Just like we think in terms of concentric layers of protection around our logical assets, we do the same with our physical ones. Whether your organization has an existing facility or is planning a new one, what is the outermost layer? It could be a fence or simply a row of concrete planters. Maybe your organization is located in a single building and the lobby is this first layer. Whatever the case, you want to balance the (oftentimes) competing needs of making the facility attractive and welcoming to legitimate visitors, while conveying the message that security is treated seriously.

Beyond the outer perimeter, you want to maintain the message that security is part of the design. Visitors should have to sign in and be escorted. All staff should wear badges that are different from badges issued to visitors. Cameras should be conspicuous throughout. “Restricted area” signs should be visible. To gain access to these restricted areas, staff should be required to badge in so that an audit record of who