11. Security

If you reveal your secrets to the wind, you should not blame the wind for revealing them to the trees.

—Kahlil Gibran

Security is a measure of the system's ability to protect data and information from unauthorized access while still providing access to people and systems that are authorized. An attack—that is, an action taken against a computer system with the intention of doing harm—can take a number of forms. It may be an unauthorized attempt to access data or services or to modify data, or it may be intended to deny services to legitimate users.

The simplest approach to characterizing security focuses on three characteristics: confidentiality, integrity, and availability (CIA):

- *Confidentiality* is the property that data or services are protected from unauthorized access. For example, a hacker cannot access your income tax returns on a government computer.
- *Integrity* is the property that data or services are not subject to unauthorized manipulation. For example, your grade has not been changed since your instructor assigned it.
- *Availability* is the property that the system will be available for legitimate use. For example, a denial-of-service attack won't prevent you from ordering *this* book from an online bookstore.

We will use these characteristics in our general scenario for security.

One technique that is used in the security domain is threat modeling. An "attack tree," which is similar to the fault tree discussed in Chapter 4, is used by security engineers to determine possible threats. The root of the tree is a successful attack, and the nodes are possible direct causes of that successful attack. Children nodes decompose the direct causes, and so forth. An attack is an attempt to compromise CIA, with the leaves of attack trees being the stimulus in the scenario. The response to the attack is to preserve CIA or deter attackers through monitoring of their activities.

Privacy

An issue closely related to security is the quality of privacy. Privacy concerns have become more important in recent years and are enshrined into law in the European Union through the General Data Protection Regulation (GDPR). Other jurisdictions have adopted similar regulations.

Achieving privacy is about limiting access to information, which in turn is about which information should be access-limited and to whom access should be allowed. The general term for information that should be kept private is personally identifiable information (PII). The National Institute of Standards and Technology (NIST) defines PII as "any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and (2) any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information."

The question of who is permitted access to such data is more complicated. Users are routinely asked to review and agree to privacy agreements initiated by organizations. These privacy agreements detail who, outside of the collecting organization, is entitled to see PII. The collecting organization itself should have policies that govern who within that organization can have access to such data. Consider, for example, a tester for a software system. To perform tests, realistic data should be used. Does that data include PII? Generally, PII is obscured for testing purposes.

Frequently the architect, perhaps acting for the project manager, is asked to verify that PII is hidden from members of the development team who do not need to have access to PII.

11.1 Security General Scenario

From these considerations, we can now describe the individual portions of a security general scenario, which is summarized in Table 11.1.

Table 11.1 *Security General Scenario*

Portion of Scenario	Description	Possible Values
Source	The attack may be from outside the organization or from inside the organization. The source of the attack may be either a human or another system. It may have been previously identified (either correctly or incorrectly) or may be currently unknown.	▪ Human ▪ Another system which is: ▪ Inside the organization ▪ Outside the organization ▪ Previously identified ▪ Unknown
Stimulus	The stimulus is an attack.	An unauthorized attempt to: ▪ Display data ▪ Capture data ▪ Change or delete data ▪ Access system services ▪ Change the system's behavior ▪ Reduce availability
Artifact	What is the target of the attack?	▪ System services ▪ Data within the system ▪ A component or resources of the system ▪ Data produced or consumed by the system
Environment	What is the state of the system when the attack occurs?	The system is: ▪ Online or offline ▪ Connected to or disconnected from a network ▪ Behind a firewall or open to a network ▪ Fully operational ▪ Partially operational ▪ Not operational

Response	The system ensures that confidentiality, integrity, and availability are maintained.	Transactions are carried out in a fashion such that ▪ Data or services are protected from unauthorized access ▪ Data or services are not being manipulated without authorization ▪ Parties to a transaction are identified with assurance ▪ The parties to the transaction cannot repudiate their involvements ▪ The data, resources, and system services will be available for legitimate use The system tracks activities within it by ▪ Recording access or modification ▪ Recording attempts to access data, resources, or services ▪ Notifying appropriate entities (people or systems) when an apparent attack is occurring
Response measure	Measures of a system's response are related to the frequency of successful attacks, the time and cost to resist and repair attacks, and the consequential damage of those attacks.	One or more of the following: ▪ How much of a resource is compromised or ensured ▪ Accuracy of attack detection ▪ How much time passed before an attack was detected ▪ How many attacks were resisted ▪ How long it takes to recover from a successful attack ▪ How much data is vulnerable to a particular attack

Figure 11.1 shows a sample concrete scenario derived from the general scenario: *A disgruntled employee at a remote location attempts to improperly modify the pay rate*

table during normal operations. The unauthorized access is detected, the system maintains an audit trail, and the correct data is restored within one day.

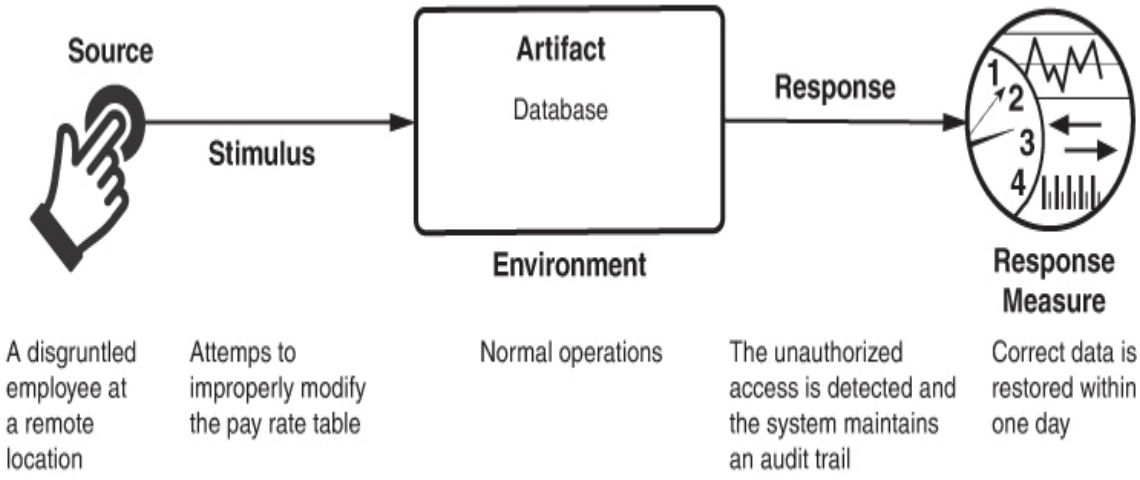

Figure 11.1 *Sample scenario for security*

11.2 Tactics for Security

One method for thinking about how to achieve security in a system is to focus on physical security. Secure installations permit only limited access to them (e.g., by using fences and security checkpoints), have means of detecting intruders (e.g., by requiring legitimate visitors to wear badges), have deterrence mechanisms (e.g., by having armed guards), have reaction mechanisms (e.g., automatic locking of doors), and have recovery mechanisms (e.g., off-site backup). These lead to our four categories of tactics: detect, resist, react, and recover. The goal of security tactics is shown in Figure 11.2, and Figure 11.3 outlines these categories of tactics.

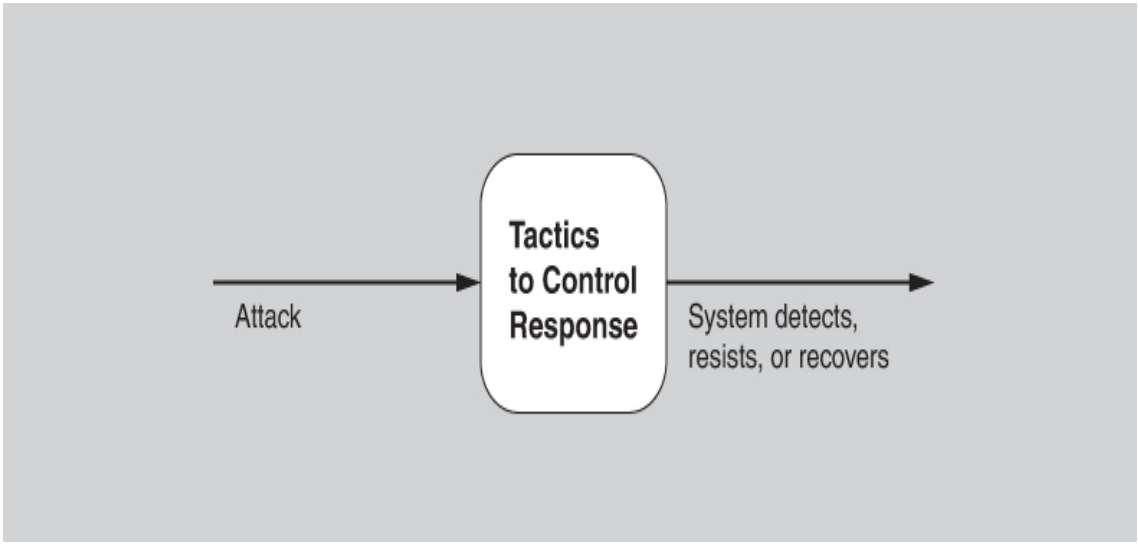

Figure 11.2 *Goal of security tactics*

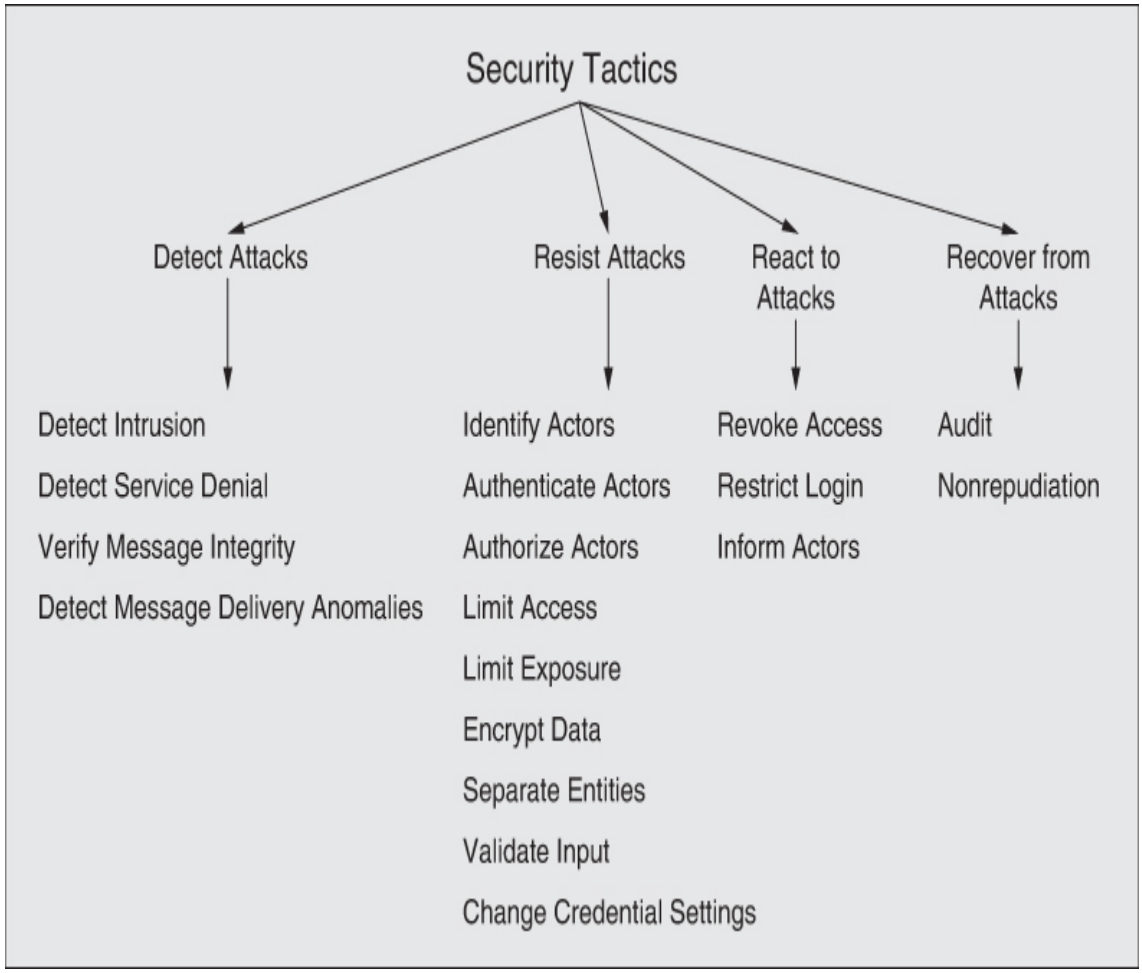

Figure 11.3 *Security tactics*

Detect Attacks

The detect attacks category consists of four tactics: detect intrusion, detect service denial, verify message integrity, and detect message delay.

- *Detect intrusion.* This tactic compares network traffic or service request patterns within a system to a set of signatures or known patterns of malicious behavior stored in a database. The signatures can be based on protocol characteristics, request characteristics, payload sizes, applications, source or destination address, or port number.
- *Detect service denial.* This tactic compares the pattern or signature of network traffic coming into a system to historical profiles of known denial-of-service (DoS) attacks.
- *Verify message integrity.* This tactic employs techniques such as checksums or hash values to verify the integrity of messages, resource files, deployment files, and configuration files. A checksum is a validation mechanism wherein the system separately maintains redundant information for files and

messages, and uses this redundant information to verify the file or message. A hash value is a unique string generated by a hashing function, whose input could be files or messages. Even a slight change in the original files or messages results in a significant change in the hash value.

- *Detect message delivery anomalies.* This tactic seeks to detect potential man-in-the-middle-attacks, in which a malicious party is intercepting (and possibly modifying) messages. If message delivery times are normally stable, then by checking the time that it takes to deliver or receive a message, it becomes possible to detect suspicious timing behavior. Similarly, abnormal numbers of connections and disconnections may indicate such an attack.

Resist Attacks

There are a number of well-known means of resisting an attack:

- *Identify actors*. Identifying actors (users or remote computers) focuses on identifying the source of any external input to the system. Users are typically identified through user IDs. Other systems may be "identified" through access codes, IP addresses, protocols, ports, or some other means.
- *Authenticate actors*. Authentication means ensuring that an actor is actually who or what it purports to be. Passwords, one-time passwords, digital certificates, two-factor authentication, and biometric identification provide a means for authentication. Another example is CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), a type of challenge–response test that is used to determine whether the user is human. Systems may require periodic reauthentication, such as when your smartphone automatically locks after a period of inactivity.
- *Authorize actors*. Authorization means ensuring that an authenticated actor has the rights to access and modify either data or services. This mechanism is usually enabled by providing some access control mechanisms within a system. Access control can be assigned per actor, per actor class, or per role.
- *Limit access*. This tactic involves limiting access to computer resources. Limiting access might mean restricting the number of access points to the resources, or restricting the type of traffic that can go through the access points. Both kinds of limits minimize the attack surface of a system. For example, a demilitarized zone (DMZ) is used when an organization wants to let external users access certain services but not access other services. The DMZ sits between the Internet and an intranet, and is protected by a pair of firewalls, one on either side. The internal firewall is a single point of access to the intranet; it functions as a limit to the number of access points as well as controls the type of traffic allowed through to the intranet.

- *Limit exposure.* This tactic focuses on minimizing the effects of damage caused by a hostile action. It is a passive defense since it does not proactively prevent attackers from doing harm. Limiting exposure is typically realized by reducing the amount of data or services that can be accessed through a single access point, and hence compromised in a single attack.
- *Encrypt data.* Confidentiality is usually achieved by applying some form of encryption to data and to communication. Encryption provides extra protection to persistently maintained data beyond that available from authorization. Communication links, by comparison, may not have authorization controls. In such cases, encryption is the only protection for passing data over publicly accessible communication links. Encryption can be symmetric (readers and writers use the same key) or asymmetric (with readers and writers use paired public and private keys).
- *Separate entities.* Separating different entities limits the scope of an attack. Separation within the system can be done through physical separation on different servers attached to different networks, the use of virtual machines, or an "air gap"—that is, by having no electronic connection between different portions of a system. Finally, sensitive data is frequently separated from nonsensitive data to reduce the possibility of attack by users who have access to nonsensitive data.
- *Validate input.* Cleaning and checking input as it is received by a system, or portion of a system, is an important early line of defense in resisting attacks. This is often implemented by using a security framework or validation class to perform actions such as filtering, canonicalization, and sanitization of input. Data validation is the main form of defense against attacks such as SQL injection, in which malicious code is inserted into SQL statements, and cross-site scripting (XSS), in which malicious code from a server runs on a client.
- *Change credential settings.* Many systems have default security settings assigned when the system is delivered. Forcing the user to change those settings will prevent attackers from gaining access to the system through settings that may be publicly available. Similarly, many systems require users to choose a new password after some maximum time period.

React to Attacks

Several tactics are intended to respond to a potential attack.

- *Revoke access.* If the system or a system administrator believes that an attack is under way, then access can be severely limited to sensitive resources, even for normally legitimate users and uses. For example, if your desktop has been

compromised by a virus, your access to certain resources may be limited until the virus is removed from your system.

- *Restrict login*. Repeated failed login attempts may indicate a potential attack. Many systems limit access from a particular computer if there are repeated failed attempts to access an account from that computer. Of course, legitimate users may make mistakes in attempting to log in, so the limited access may last for only a certain time period. In some cases, systems double the lock-out time period after each unsuccessful login attempt.
- *Inform actors*. Ongoing attacks may require action by operators, other personnel, or cooperating systems. Such personnel or systems—the set of relevant actors—must be notified when the system has detected an attack.

Recover from Attacks

Once a system has detected and attempted to resist an attack, it needs to recover. Part of recovery is restoration of services. For example, additional servers or network connections may be kept in reserve for such a purpose. Since a successful attack can be considered a kind of failure, the set of availability tactics (from Chapter 4) that deal with recovering from a failure can be brought to bear for this aspect of security as well.

In addition to the availability tactics for recovery, the audit and nonrepudiation tactics can be used:

- *Audit.* We audit systems—that is, keep a record of user and system actions and their effects—to help trace the actions of, and to identify, an attacker. We may analyze audit trails to attempt to prosecute attackers, or to create better defenses in the future.
- *Nonrepudiation.* This tactic guarantees that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message. For example, you cannot deny ordering something from the Internet, and the merchant cannot disclaim getting your order. This could be achieved with some combination of digital signatures and authentication by trusted third parties.

11.3 Tactics-Based Questionnaire for Security

Based on the tactics described in Section 11.2, we can create a set of security tactics–inspired questions, as presented in Table 11.2. To gain an overview of the architectural choices made to support security, the analyst asks each question and records the answers in the table. The answers to these questions can then be made the focus of further activities: investigation of documentation, analysis of code or other artifacts, reverse engineering of code, and so forth.

Table 11.2 *Tactics-Based Questionnaire for Security*

Tactics Group	Tactics Question	Supported? (Y/N)	Risk	Design Decisions and Location	Rationale and Assumptions
Detecting Attacks	Does the system support the **detection of intrusions** by, for example, comparing network traffic or service request patterns within a system to a set of signatures or known patterns of malicious behavior stored in a database?				
	Does the system support the **detection of denial-of-service attacks** by, for example, comparing the pattern or signature of network traffic coming into a system to historical profiles of known DoS attacks?				
	Does the system support the **verification of message integrity** via techniques such as checksums or hash values?				

	Does the system support the **detection of message delays** by, for example, checking the time that it takes to deliver a message?				
Resisting Attacks	Does the system support the **identification of actors** through user IDs, access codes, IP addresses, protocols, ports, etc.?				
	Does the system support the **authentication of actors** via, for example, passwords, digital certificates, two-factor authentication, or biometrics?				

	Does the system support the **authorization of actors**, ensuring that an authenticated actor has the rights to access and modify either data or services?				
	Does the system support **limiting access** to computer resources via restricting the number of access points to the resources, or restricting the type of traffic that can go through the access points?				
	Does the system support **limiting exposure** by reducing the amount of data or services that can be accessed through a single access point?				
	Does the system support **data encryption**, for data in transit or data at rest?				

	Does the system design consider the **separation of entities** via physical separation on different servers attached to different networks, virtual machines, or an “air gap”?				
	Does the system support **changing credential settings**, forcing the user to change those settings periodically or at critical events?				
	Does the system **validate input** in a consistent, system-wide way—for example, using a security framework or validation class to perform actions such as filtering, canonicalization, and sanitization of external input?				
Reacting to Attacks	Does the system support **revoking access** by limiting access to sensitive resources, even for normally legitimate users and uses if an attack is under way?				

	Does the system support **restricting login** in instances such as multiple failed login attempts?				
	Does the system support **informing actors** such as operators, other personnel, or cooperating systems when the system has detected an attack?				
Recovering from Attacks	Does the system support maintaining an **audit** trail to help trace the actions of, and to identify, an attacker?				
	Does the system guarantee the property of **nonrepudiation**, which guarantees that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message?				
	Have you checked the "recover from faults" category of tactics from Chapter 4?				

11.4 Patterns for Security

Two of the more well-known patterns for security are intercepting validator and intrusion prevention system.

Intercepting Validator

This pattern inserts a software element—a wrapper—between the source and the destination of messages. This approach assumes greater importance when the source of the messages is outside the system. The most common responsibility of this pattern is to implement the verify message integrity tactic, but it can also incorporate tactics such as detect intrusion and detect service denial (by comparing messages to known intrusion patterns), or detect message delivery anomalies.

Benefits:

- Depending on the specific validator that you create and deploy, this pattern can cover most of the waterfront of the "detect attack" category of tactics, all in one package.

Tradeoffs:

- As always, introducing an intermediary exacts a performance price.
- Intrusion patterns change and evolve over time, so this component must be kept up-to-date so that it maintains its effectiveness. This imposes a maintenance obligation on the organization responsible for the system.

Intrusion Prevention System

An intrusion prevention system (IPS) is a standalone element whose main purpose is to identify and analyze any suspicious activity. If the activity is deemed acceptable, it is allowed. Conversely, if it is suspicious, the activity is prevented and reported. These systems look for suspicious patterns of overall usage, not just anomalous messages.

Benefits:

- These systems can encompass most of the "detect attacks" and "react to attacks" tactics.

Tradeoffs:

- The patterns of activity that an IPS looks for change and evolve over time, so the patterns database must be constantly updated.
- Systems employing an IPS incur a performance cost.

- IPSs are available as commercial off-the-shelf components, which makes them unnecessary to develop but perhaps not entirely suited to a specific application.

Other notable security patterns include compartmentalization and distributed responsibility. Both of these combine the "limit access" and "limit exposure" tactics—the former with respect to information, the latter with respect to activities.

Just as we included (by reference) tactics for availability in our list of security tactics, patterns for availability also apply to security by counteracting attacks that seek to stop the system from operating. Consider the availability patterns discussed in Chapter 4 here as well.

11.5 For Further Reading

The architectural tactics that we have described in this chapter are only one aspect of making a system secure. Other aspects include the following:

- Coding. *Secure Coding in C and C++*[Seacord 13] describes how to code securely.
- Organizational processes. Organizations must have processes that take responsibility for various aspects of security, including ensuring that systems are upgraded to put into place the latest protections. NIST 800-53 provides an enumeration of organizational processes [NIST 09]. Organizational processes must account for insider threats, which account for 15–20 percent of attacks. [Cappelli 12] discusses insider threats.
- Technical processes. Microsoft's Security Development Lifecycle includes modeling of threats: microsoft.com/download/en/details.aspx?id=16420.

The Common Weakness Enumeration is a list of the most common categories of vulnerabilities discovered in systems, including SQL injection and XSS: https://cwe.mitre.org/.

NIST has published several volumes that give definitions of security terms [NIST 04], categories of security controls [NIST 06], and an enumeration of security controls that an organization could employ [NIST 09]. A security control could be a tactic, but it could also be organizational, coding, or technical in nature.

Good books on engineering systems for security include Ross Anderson's *Security Engineering: A Guide to Building Dependable Distributed Systems*, third edition [Anderson 20] and the series of books by Bruce Schneier.

Different domains have different sets of security practices that are relevant to their domain. The Payment Card Industry (PCI), for example, has established a set

of standards intended for those involved in credit card processing (pcisecuritystandards.org/).

The Wikipedia page on "Security Patterns" contains brief definitions of a large number of security patterns.

Access control is commonly performed using a standard called OAuth. You can read about OAuth at https://en.wikipedia.org/wiki/OAuth.

11.6 Discussion Questions

1. Write a set of concrete scenarios for security for an automobile. Consider in particular how you would specify scenarios regarding control of the vehicle.
2. One of the most sophisticated attacks on record was carried out by a virus known as Stuxnet. Stuxnet first appeared in 2009, but became widely known in 2011 when it was revealed that it had apparently severely damaged or incapacitated the high-speed centrifuges involved in Iran's uranium enrichment program. Read about Stuxnet and see if you can devise a defense strategy against it based on the tactics described in this chapter.
3. Security and usability are often seen to be at odds with each other. Security often imposes procedures and processes that seem like needless overhead to the casual user. Nevertheless, some say that security and usability go (or should go) hand in hand, and argue that making the system easy to use securely is the best way to promote security to the users. Discuss.
4. List some examples of critical resources for security, which a DoS attack might target and try to exhaust. Which architectural mechanisms could be employed to prevent this kind of attack?
5. Which of the tactics detailed in this chapter will protect against an insider threat? Can you think of any that should be added?
6. In the United States, Netflix typically accounts for more than 10 percent of all Internet traffic. How would you recognize a DoS attack on Netflix.com? Can you create a scenario to characterize this situation?
7. The public disclosure of vulnerabilities in an organization's production systems is a matter of controversy. Discuss why this is so, and identify the pros and cons of public disclosure of vulnerabilities. How could this issue affect your role as an architect?
8. Similarly, the public disclosure of an organization's security measures and the software to achieve them (via open source software, for example) is a matter of controversy. Discuss why this is so, identify the pros and cons of public disclosure of security measures, and describe how this could affect your role as an architect.