CHAPTER 3

Authorization and Access Control

Information in This Chapter:

- Authorization
- Access Control
- Access Control Methodologies

INTRODUCTION

Once we have received a claim of identity and established whether that claim is valid, as we discussed in Chapter 2, we move on to what the party is allowed to do and whether we will allow or deny them access to our resources. We can achieve this with two main concepts: authorization and access control. Authorization allows us to specify where the party should be allowed or denied access, and access control enables us to manage this access at a very granular level.

Access controls can be constructed in a variety of manners. We can base access controls on physical attributes, sets of rules, lists of individuals or systems, or more complex factors. The particular type of access control often depends on the environment in which it is to be used. We can find simpler access controls implemented in many applications and operating systems, while more complex multilevel configurations might be implemented in military or government environments. In such cases, the importance of what we are controlling access to may dictate that we track what our users have access to across a number of levels of sensitivity.

When we discuss access control concepts, we may be referring to them in a purely logical or physical sense or, more commonly, as a combination of the two. In terms of access control systems, it is important to understand that, when dealing with computing environments, the logical and physical are often closely entangled. Logical access control systems, even those that do not have an immediately obvious physical component, are still dependent on physical

hardware, networks, and utilities to carry out their tasks. Likewise, many, but not all, physical access controls (sometimes referred to as guards, gates, and guns) have some sort of logical component. Often the systems that control our access to and within facilities depend equally on networks, computer systems, and other similar components. In many ways, information security and physical security are closely linked to each other.

AUTHORIZATION

Authorization is the next step taken after we have completed identification and authentication, as shown in Figure 3.1. Authorization enables us to determine, once we have authenticated the party in question, exactly what they are allowed to do. We typically implement authorization through the use of access controls, which we will discuss later in this chapter.

Principle of Least Privilege

When we are determining what access we will provide to the parties to whom we have provided authorized access, there is an important concept we should keep in mind, called the principle of least privilege. The principle of least privilege dictates that we should only allow the bare minimum of access to a party—this might be a person, user account, or process—to allow it to perform the functionality needed of it. For example, someone working in a sales department should not need access to data in our internal human resources system in order to do their job. Violation of the principle of least privilege is the heart of many of the security problems we face today.

One of the more common ways in which we find the principle of least privilege improperly implemented is in the permissions we give operating system (OS) user accounts, most commonly violated by users and administrators of Microsoft operating systems. In Microsoft operating systems, we will often find that casual users of the OS, who are performing tasks such as creating documents in word processors and exchanging e-mail, are configured with administrative access, thus allowing them to carry out any task that the OS allows. As a consequence of this, whenever the overprivileged user opens an e-mail attachment containing malware, or encounters a Web site that pushes attack code to the client computer, these attacks have free reign on the system because they are acting as the user, who is, in turn, endowed with administrative capabilities. Because of this, the attacker's job is much easier, as they can simply turn

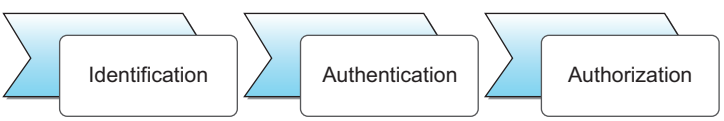

FIGURE 3.1
Identification, Authentication, and Authorization

off anti-malware tools, install any additional attack tools they care to, and proceed with completely compromising the system.

We can see the same issue in services or processes that are running at a more privileged level than they need to in order to carry out their functions. If we have a service running a Web server, for instance, this service only needs sufficient permission to access the files and scripts that directly pertain to the Web content it is serving, and nothing more. If we allow the Web service to access additional files in the file system, an attacker could potentially read or alter these files to gain unauthorized access to more sensitive information than we would normally make public, thus giving the attacker an inroad to attack deeper into the system.

By carefully following the principle of least privilege when configuring systems, allocating permissions for accounts, and planning out our security, we can take away some of the more easily accessed tools that attackers can use against us. This is a very simple security measure that we can put in place, and it is very effective.

ACCESS CONTROL

When we look at access controls we have four basics tasks we might want to carry out: allowing access, denying access, limiting access, and revoking access. Among these four actions, we can describe most access control issues or situations.

Allowing access lets us give a particular party, or parties, access to a given resource. For example, we might want to give a particular user access to a file, or we may want to give an entire group of people access to all the files in a given directory. We might also be referring to access in a physical sense, by giving our employees access to our facility through the use of a key or badge.

Denying access is the diametric opposite of granting access. When we deny access we are preventing access by a given party to the resource in question. We might be denying access to a particular person attempting to log on to a machine based on the time of day, or we might deny unauthorized individuals from entering the lobby of our building beyond business hours. Many access control systems are set to deny by default, with the authorized users only being permitted access.

Limiting access refers to allowing some access to our resource, but only up to a certain point. This is very important when we are using applications that may be exposed to attack-prone environments, as we see with Web browsers used on the Internet. In such cases, we might see the application being run in a sandbox in order to limit what can be done outside the context of the application. In a physical sense, we can see the concept of access control limitations in the different levels of keying that we might see in the locks in a building. We may have a master key that can open any door in the building, an intermediate key that can open only a few doors, and a low-level key that can open only one door.

MORE ADVANCED

When we look at limiting the access for software, we will often see the term *sandbox* used to describe the limitations that are put in place. A sandbox is simply a set of resources devoted to a program, process, or similar entity, outside of which the entity cannot operate. We use sandboxes to prevent their contents from accessing files, memory, and other system resources with which they should not be interacting. Sandboxes can be very useful for containing things that we cannot trust, such as code from public Web sites. We can see an example of a sandbox in use in the Java Virtual Machine (JVM) under which programs written in the Java programming language run. The JVM is specifically constructed to protect users against potentially malicious software that they might download.

Revocation of access is a very important idea in access control. It is vital that once we have given a party access to a resource, we be able to take that access away again. If we were, for instance, to fire an employee, we would want to revoke any accesses that they might have. We would want to remove access to their e-mail account, disallow them from connecting to our virtual private network (VPN), deactivate their badge so that they can no longer enter the facility, and revoke other accesses that they might have. Particularly when we are working with computer-oriented resources in some fashion, it may be vital to be able to revoke access to a given resource very quickly.

When we look to implement access controls, there are two main methods that we might use: access control lists and capabilities. Each of these has positives and negatives, and the ways we can carry out the four basic tasks we covered earlier will differ depending on the method we choose for our access control implementation.

Access Control Lists

Access control lists (ACLs), often referred to as "ackles," are a very common choice of access control implementation. ACLs are usually used to control access in the file systems on which our operating systems run and to control the flow of traffic in the networks to which our systems are attached.

When ACLs are constructed, they are typically built specifically to a certain resource, and they contain the identifiers of the party allowed to access the resource in question and what the party is allowed to do in relation to the resource. As we see in Figure 3.2, Alice is allowed access to the resource, while Bob is specifically denied access. This may seem like a very simplistic concept, but in the context of larger ACL implementations, such as those used in file systems, ACLs can become quite complex.

FILE SYSTEM ACLs

When we look at the ACLs in most file systems, we commonly see three permissions in use: read, write, and execute, respectively allowing us to access the

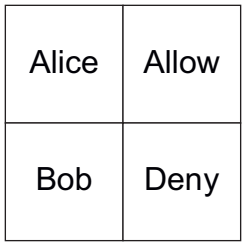

FIGURE 3.2
A Simple ACL

contents of a file or directory, write to a file or directory, and, presuming that a file contains either a program or a script capable of running on the system in question, execute the contents of the file.

In the case of file systems, a file or directory may also have multiple ACLs attached to it. In UNIX-like operating systems, for instance, we can see separate access lists for a given file, in the form of *user*, *group*, and *other* ACLs. We can give an individual user read, write, and execute permissions, a group of users different read, write, and execute permissions, and a different set of read, write, and execute permissions to anyone that is not an individual or group that we have already covered. These three sets of permissions will display as *rwxrwxrwx*, with the first *rwx* set representing the *user*, the second the *group*, and the third *other*, as shown in Figure 3.3.

MORE ADVANCED

To further explore the idea, we can look at the specific example of one of the files shown in Figure 3.3. If we look at the first file, .nano_history, we can see that the permissions are displayed as - *r w* - - - - - - -. This may seem a bit cryptic, but we can help this somewhat by segmenting this into the relevant sections. If we divide it as - | *r w* - | - - - | - - -, we can see where the different sections lie. The first - is generally used to represent the file type. In the case of our example, - represents a regular file, and *d* represents a directory. The second segment, the user permissions, is set to *r w* -, meaning that the user that owns the file can read it and write it, but not execute it. The third segment, the group permissions, is set to - - -, meaning that the members of the group that owns the file cannot read it, write it, or execute it. The last segment is also set to - - -, meaning that anyone that is not the user that owns the file or in the group that owns the file can also not read, write, or execute it.

By using such sets of file permissions, we can, in a simple fashion, control access to the operating systems and applications that utilize our file system. Although we only looked at file system permissions like these as they pertain to file systems used in Microsoft and UNIX-like operating systems, most file systems use a very similar, if not identical, set of permissions.

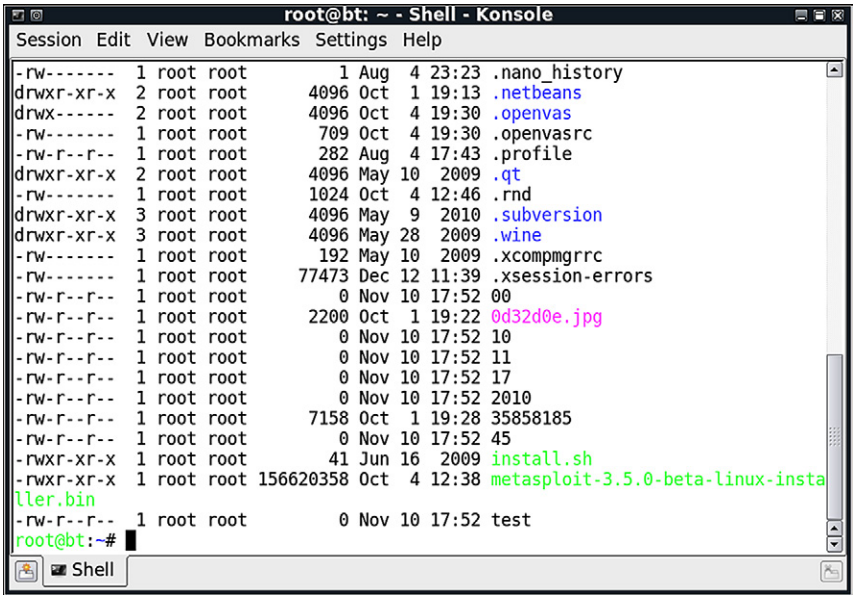

FIGURE 3.3
File Permissions on a Linux Operating System

NETWORK ACLs

When we look at the variety of activities that take place on networks, both private and public, we can again see ACLs regulating such activity. In the case of network ACLs, we typically see access controlled by the identifiers we use for network transactions, such as Internet Protocol (IP) addresses, Media Access Control (MAC) addresses, and ports. We can see such ACLs at work in network infrastructure such as routers, switches, and firewall devices, as well as in software firewalls, Facebook, Google, e-mail, or other forms of software.

Permissions in network ACLs tend to be binary in nature, generally consisting of allow and deny. When we set up the ACL, we use our chosen identifier or identifiers to dictate which traffic we are referring to and simply state whether the traffic is to be allowed or not.

One of the simplest forms of network-oriented ACLs that we might see in place is MAC address filtering. MAC addresses are, in theory, unique identifiers attached to each network interface in a given system. Each network interface has a hardcoded MAC address issued when it is created.

ALERT!

Unfortunately for those of us depending on MAC addresses as a basis for our ACLs, the MAC address used by a network interface can be overridden by software settings in most operating systems. Such changes are very trivial to put in place, and the MAC address is not a good choice for a unique identifier of a particular device on the network.

We can also choose to use IP addresses as the basis for filtering in our ACL. We can implement such filtering based on individual addresses, or on an entire range of IP addresses. Unfortunately, similar to the issue with using MAC addresses for ACLs, IP addresses can be falsified and are not unique to a particular network interface. Additionally, IP addresses issued by Internet service providers (ISPs) are subject to frequent change, making IP addresses as the sole basis for filtering a shaky prospect, at best.

MORE ADVANCED

Some organizations, such as those that operate Web servers, mail servers, and other services that are exposed to the Internet, apply large-scale filtering in order to block out known attacks, spammers, and other undesirable traffic. Such filtering can take the form of dropping traffic from individual IP addresses, to ranges, to the entire IP space of large organizations, ISPs, or even entire countries. This practice is commonly referred to as blackholing, because any traffic to such filtered destinations is simply dropped and appears to have vanished into a black hole from the perspective of the sender.

We can also filter by the port being used to communicate over the network. Many common services and applications use specific ports to communicate over networks. For instance, FTP uses ports 20 and 21 to transfer files, Internet Message Access Protocol (IMAP) uses port 143 for managing e-mail, Secure Shell (SSH) uses port 22 to manage remote connections to systems, and many more—65,535 ports in all. We can control the use of many applications over the network by allowing or denying traffic originating from or sent to any ports that we care to manage. Like MAC and IP addresses, the specific ports that are used for applications are a convention, not an absolute rule. We can, with relative ease, change the ports that applications use to different ports entirely.

Using single attributes to construct ACLs is likely to present a variety of issues, including our attribute not being guaranteed to be unique, such as an IP address, or being easy to alter, such as a MAC address. When we use several attributes in combination we begin to arrive at a more secure technique. A very commonly used combination is that of IP address and port, typically referred to as a socket. In this way, we can allow or deny network traffic from one or more IP addresses using one or more applications on our network in a workable fashion.

We can also construct ACLs to filter on a wide variety of other things. In some cases, we might want to monitor the traffic going over our network in order to allow or deny traffic based on more specific criteria, such as the content of an individual packet or a related series of packets. Using such techniques, we can filter out traffic related to attacks, or traffic that is simply undesirable to us, such as that related to the peer-to-peer file-sharing networks commonly used to illegally share copyrighted songs, videos, and software.

Capabilities

Capability-based security can provide us with an alternate solution to access control that uses a different structure than what we see in ACLs. Where ACLs define the permissions based on a given resource, an identity, and a set of permissions, all generally held in a file of some sort, capabilities are oriented around the use of a token that controls our access. We can think of a token in a capability as being analogous to the badge we might use to open the door in a building. We have one door, and many people have a token that will open it, but we can have differing levels of access. Where one person might be able to access the building only during business hours on weekdays, another person may have permission to enter the building at any time of day on any day of the week.

Interestingly, in capability-based systems, the right to access a resource is based entirely on possession of the token, and not who possesses it. As with our badge example, if we were to give our badge to someone else, he would be able to use it to access the building with whatever set of permissions we have. In a capability-based system, applications can share with other applications the token that defines their level of access. In noncapability-based systems, which use ACLs to manage permissions, we may experience the confused deputy problem, due to the way that access control is implemented.

CONFUSED DEPUTY PROBLEM

The confused deputy problem is a type of attack that is common in systems that use ACLs rather than capabilities. The crux of the confused deputy problem is seen when the software with access to a resource has a greater level of permission to access the resource than the user who is controlling the software. If we, as the user, can trick the software into misusing its greater level of authority, we can potentially carry out an attack [1]. We will discuss a few practical examples of attacks that exploit the confused deputy problem later in this section.

Several specific attacks, many of them client side in nature, can take practical advantage of the confused deputy problem. These often involve tricking the user into taking some action when they really think they are doing something else entirely. Two of the more common uses of such an attack are client-side attacks such as cross-site request forgery (CSRF) and clickjacking.

Client-side attacks are attacks that take advantage of weaknesses in applications that are running on the computer being operated directly by the user, often referred to as the client. These attacks can take the form of code sent through the Web browser, which is then executed on the local machine, malformed PDF files, images or videos with attack code embedded, or other forms. In the past several years, software vendors have become more aware of such attacks as an issue and have begun building defensive measures into their software, but new attacks appear on a regular basis.

CSRF is an attack that misuses the authority of the browser on the user's computer. If the attacker knows of, or can guess, a Web site to which the user might already be authenticated, perhaps a very common site such as Amazon.com, he can attempt to carry out a CSRF attack [2]. He can do this by embedding a link in a Web page or HTML-based e-mail, generally a link to an image from the site to which he wishes to direct the user without their knowledge. When the application attempts to retrieve the image in the link, it also executes the additional commands the attacker has embedded in it. In our example, when the user's browser loads the image from Amazon.com, as long as the authentication cookie for Amazon has not expired, the attacker might cause the user to make a purchase without their knowledge, thus allowing the attacker to sell more copies of a book.

Clickjacking, also known as user interface redressing, is a particularly sneaky and effective client-side attack that takes advantage of some of the page rendering features that are available in newer Web browsers. In order to carry out a clickjacking attack, the attacker must legitimately control or have taken control of some portion of the Web site that is to be used as the attack vehicle. The attacker constructs or modifies the site in order to place an invisible layer over something the client would normally click on, in order to cause the client to execute a command differing from what they actually think they are performing [3]. Clickjacking can be used to trick the client into making purchases, changing permissions in their applications or operating systems, or performing other nefarious activities.

If we were to use capabilities instead of ACLs to manage permissions, these attacks would not be possible. In the case of each of these attacks, the misuse of permissions would not be possible, because the attacker would not be able to misuse the authority of the user without actually having access to the token that would allow him permission to do so.

ALERT!

Browser attacks are very common and are likely to succeed against systems that have not been hardened against them specifically. Some of the more commonly used browsers, such as Microsoft's Internet Explorer and Mozilla Firefox, now include at least a rudimentary form of protection against such attacks. Browser security plug-ins, such as NoScript[A] for Firefox and GuardedID[B] for Internet Explorer, can also help to foil such attacks.

[A] http://noscript.net/
[B] www.guardedid.com/default.aspx

Unfortunately, the most commonly used operating systems have only a very minimal implementation of capability-based security, and this does not often extend to the sharing of permissions between applications. In most cases, in

order to mitigate the attacks that we discussed, additional layers of security, in the form of applications or plug-ins, are needed.

ACCESS CONTROL METHODOLOGIES

Access controls are the means by which we implement authorization and deny or allow access to parties, based on what resources we have determined they should be allowed access to. Although the term may sound very technical and oriented in the direction of high-security computing facilities, access controls are something we deal with on a daily basis.

When we lock or unlock the doors on our house, we are using a form of physical access control, based on the keys (something you have) that we use.

When we start our car, we are also likely to use a key. For some newer cars, our key may even include an extra layer of security by adding Radio Frequency Identification (RFID) tags, certificate-like identifiers stored on the key itself, and other security technologies.

Upon reaching our place of employment, we might use a badge (something you have) to enter the building, once again, a physical access control.

When we sit down in front of our computer at work and type in our password (something you know), we are authenticating and using a logical access control system in order to access the resources to which we have been given permission. Depending on the environments we pass through in the course of working, going to school, and performing the other activities that make up our day, we may have more or less exposure to access controls, but most of us see multiple implementations like these on a regular basis.

Access Control Models

There are quite a few different access control models we might run across, and we will cover the most common models here. The most likely set we will encounter in the security world includes discretionary access control, mandatory access control, rule-based access control, role-based access control, and attribute-based access control.

DISCRETIONARY ACCESS CONTROL

Discretionary access control (DAC) is a model of access control based on access being determined by the owner of the resource in question. The owner of the resource can decide who does and does not have access, and exactly what access they are allowed to have. In Microsoft operating systems, we can see DAC implemented. If we decide to create a network share, for instance, we get to decide who we want to allow access.

MANDATORY ACCESS CONTROL

Mandatory access control (MAC) is a model of access control in which the owner of the resource does not get to decide who gets to access it, but instead

access is decided by a group or individual who has the authority to set access on resources. We can often find MAC implemented in government organizations, where access to a given resource is largely dictated by the sensitivity label applied to it (secret, top secret, etc.), by the level of sensitive information the individual is allowed to access (perhaps only secret), and by whether the individual actually has a need to access the resource, as we discussed when we talked about the principle of least privilege earlier in this chapter.

MORE ADVANCED

It is worthwhile to note that MAC is an overloaded acronym, in that it can have more than one meaning. In this case, two of the more common meanings are Media Access Control (MAC), as in the unique identifier for a network interface, and mandatory access control (MAC), in the sense of a type of access control.

ROLE-BASED ACCESS CONTROL

Role-based access control (RBAC) is a model of access control that, similar to MAC, functions on access controls set by an authority responsible for doing so, rather than by the owner of the resource. The difference between RBAC and MAC is that access control in RBAC is based on the role the individual being granted access is performing. For example, if we have an employee whose only role is to enter data into a particular application, through RBAC we would only allow the employee access to that application, regardless of the sensitivity or lack of sensitivity of any other resource he might potentially access. If we have an employee with a more complex role—customer service for an online retail application, perhaps—the employee's role might require him to have access to information about customers' payment status and information, shipping status, previous orders, and returns, in order to be able to assist said customers. In this case, RBAC would grant him considerably more access. We can see RBAC implemented in many large-scale applications that are oriented around sales or customer service.

ATTRIBUTE-BASED ACCESS CONTROL

Attribute-based access control (ABAC) is, logically, based on attributes. These can be the attributes of a particular person, of a resource, or of an environment.

Subject attributes are those of a particular individual. We could choose any number of attributes, such as the classic "you must be this tall to ride" access control, which exists to prevent the altitudinally challenged from riding on amusement park rides that might be harmful to them. Another very common example can be seen in the use of a Captcha, as shown in Figure 3.4. Captchas are used to control access, based on whether the party on the other end can pass a test that is, in theory, too difficult for a machine to complete, thus proving the party to be human. Captcha or, more properly, CAPTCHA, stands for Completely Automated Public Turing Test to Tell Humans and Computers Apart [4].

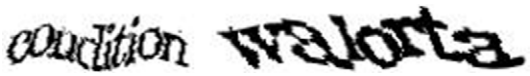

FIGURE 3.4
A Captcha

Resource attributes are those that relate to a particular resource, such as an operating system or application. We often see this occur, although usually for technical reasons rather than security reasons, when we encounter software that only runs on a particular operating system, or Web sites that only work with certain browsers. We might apply this type of access control as a security measure by requiring specific software to be used or particular protocols for communication.

Environmental attributes can be used to enable access controls that operate based on environmental conditions. We commonly use the time attribute to control access, in both a physical and a logical sense, based on length of time passed, or time of day. Access controls on buildings are often configured to only allow access during certain hours of the day, such as during business hours. We also see time limits set on VPN connections, forcing the user to reconnect every 24 hours. This is often done to prevent users from keeping such a connection running after their authorization for using it has been removed. We can often find ABAC implemented on infrastructure systems such as those in network or telecommunications environments.

MULTILEVEL ACCESS CONTROL

Multilevel access control models are used where the simpler access control models that we just discussed are considered to not be robust enough to protect the information to which we are controlling access. Such access controls are used extensively by military and government organizations, or those that often handle data of a very sensitive nature. We might see multilevel security models used to protect a variety of data, from nuclear secrets to protected health information (PHI).

The Bell-LaPadula model implements a combination of DAC and MAC access controls, and is primarily concerned with the confidentiality of the resource in question. Generally, in cases where we see DAC and MAC implemented together, MAC takes precedence over DAC, and DAC works within the accesses allowed by the MAC permissions. For example, we might have a resource that is classified as secret and a user that has a secret level of clearance, normally allowing them to access the resource under the accesses allowed by MAC. However, we might also have an additional layer of DAC under the MAC access, and if the resource owner has not given the user access, they would not be able to access it, despite the MAC permissions. In Bell-LaPadula, we have two security properties that define how information can flow to and from the resource [5]:

- *The Simple Security Property*: The level of access granted to an individual must be at least as high as the classification of the resource in order for the individual to be able to access it.

- *The *Property*: Anyone accessing a resource can only write its contents to one classified at the same level or higher.

These properties are generally summarized as "no read up" and "no write down," respectively. In short, this means that when we are handling classified information, we cannot read any higher than our clearance level, and we cannot write classified data down to any lower level.

The Biba model of access control is primarily concerned with protecting the integrity of data, even at the expense of confidentiality. Biba has two security rules that are the exact reverse of those we discussed in the Bell-LaPadula model [6]:

- *The Simple Integrity Axiom:* The level of access granted to an individual must be no lower than the classification of the resource.
- *The *Integrity Axiom:* Anyone accessing a resource can only write its contents to one classified at the same level or lower.

We can summarize these rules as "no read down" and "no write up," respectively. This may seem completely counterintuitive when we consider protecting information, but remember that we have changed the focus from confidentiality to integrity. In this case, we are protecting integrity by ensuring that our resource can only be written to by those with a high level of access and that those with a high level of access do not access a resource with a lower classification.

The Brewer and Nash model, also known as the Chinese Wall model, is an access control model designed to prevent conflicts of interest. Brewer and Nash is commonly used in industries that handle sensitive data, such as that found in the financial, medical, or legal industry. Three main resource classes are considered in this model [7]:

- *Objects*: Resources such as files or information, pertaining to a single organization.
- *Company groups*: All objects pertaining to a particular organization.
- *Conflict classes*: All groups of objects that concern competing parties.

If we look at the example of a commercial law firm working for companies in a certain industry, we might have files that pertain to various individuals and companies working in that industry. As an individual lawyer at the firm accesses data and works for different clients, he could potentially access confidential data that would generate a conflict of interest in him while working on a new case. In the Brewer and Nash model, the resources and case materials that the lawyer was allowed access to would dynamically change based on the materials he had previously accessed.

Physical Access Controls

Many of the access control methods we have discussed throughout the chapter can be applied to physical security as well as logical security. When concerned

with physical access controls, we are often largely concerned with controlling the access of individuals and vehicles.

Access control for individuals often revolves around controlling movement into and out of buildings or facilities. We can see simple examples of such controls on the buildings of many organizations in the form of badges that moderate opening doors into or within the facility (something you have, from Chapter 2). Such badges are typically configured on an ACL that permits or denies their use for certain doors and regulates the time of day that they can be used.

One of the more common issues with physical access controls is that of tailgating. Tailgating occurs when we authenticate to the physical access control measure, such as using a badge, and then another person follows directly behind us without authenticating themselves. Tailgating can cause a variety of issues, including allowing unauthorized individuals into the building and creating an inaccurate representation of who is actually in the building in case there is an emergency.

We can attempt to solve tailgating in a variety of ways, from implementing policy that forbids doing so, to posting a guard in the area, to simply (but expensively) installing a physical access control solution that only allows one person to pass through at a time, such as a turnstile. All of these are reasonable solutions, but, depending on the environment in question, may or may not be effective. We will often find that a combination of several solutions is needed to develop a thorough and complete solution.

A much more complex example of this type of access control that many people are familiar with is the security system in use at many airports. Particularly after the terrorist attacks of 9/11 in the United States, we have seen the level of security at airports increase, much of it oriented in the direction of access controls. Once we have entered the airport security system, we are required to present a boarding pass and identification (something you have, times two). We are then typically passed through a number of steps to ensure that we do not carry any dangerous devices, a form of attribute-based access control. We then proceed to our gate and, once again, present our boarding pass to step onto the airplane. Such processes may differ slightly depending on the country in which we travel, but they are generally the same from an access control perspective.

Physical access control for vehicles often revolves around keeping said vehicles from moving into or through areas in which we do not desire them to be. This is often done through the use of various simple barriers, including Jersey barriers such as those shown in Figure 3.5, bollards, one-way spike strips, fences, and similar tools. We may also see more complex installations that include manned or unmanned rising barriers, automated gates or doors, and other similar items.

There are, of course, a huge number of other physical access controls and methods that we have not discussed here. Additionally, when we refer to physical access control devices, or access controls in general, the line between what is an authentication device and an access control device often becomes rather blurry.

FIGURE 3.5
A Jersey Barrier [8]

AUTHORIZATION AND ACCESS CONTROL IN THE REAL WORLD

We can see authorization and access control used in our personal and business lives on an almost constant basis, although the portions of these that are immediately visible to us are the access controls. Looking specifically at logical access controls, we can see them used when we log in to computers or applications, when we send traffic over the Internet, when we watch cable or satellite television, when we make a call on our mobile phones, and in thousands of other places. In some cases, such measures are visible to us and require us to enter a password or a PIN, but a large portion of them happen in the background, completely invisible to the tasks we are carrying out and taken care of by the technologies that facilitate our tasks.

In the sense of physical access controls, we see these rather frequently as well, although it may not register to us that we are seeing them. Most of us carry around a set of keys that allow us access to our homes, cars, and other devices, and these are the credentials for access to them. Many of us also carry proximity badges that allow us access to our places of employment, schools, and other areas. We can also see the access controls that manage the movement of vehicles in everyday use in vehicle-oriented areas such as parking garages and parking areas at airports, and in the vicinity of high-security areas such as the White House in the United States.

SUMMARY

Authorization is a key step in the process that we work through in order to allow entities access to resources, namely, identification, authentication, and authorization, in that order. We implement authorization through the use of access controls, more specifically through the use of access control lists and capabilities, although the latter are often not completely implemented in most of the common operating systems in use today.

The specifics of access control are defined through the various models we use when putting together such systems. We often see the use of the simpler access

control models such as discretionary access control, mandatory access control, role-based access control, and attribute-based access control in our daily lives. In environments that handle more sensitive data, such as those involved in the government, military, medical, or legal industry, we may see the use of multilevel access control models, including Bell LaPadula, Biba, Clark-Wilson, and Brewer and Nash.

Access control concepts in general largely apply to both logical and physical areas, but we do see some specialized applications when looking specifically at physical access control. Here we have several sets of access controls that apply to ensuring that people and vehicles are restricted from exiting or entering areas where they are not authorized to be. We can see examples of such controls in our daily lives at office buildings, parking areas, and high-security facilities in general.

EXERCISES

1. Discuss the difference between authorization and access control.
2. What does the Clark-Wilson model protect against?
3. Why does access control based on the MAC address of the systems on our network not represent strong security?
4. Which should take place first, authorization or authentication?
5. What are the differences between MAC and DAC in terms of access control?
6. The Bell-LaPadula and Biba multilevel access control models each have a primary security focus. Can these two models be used in conjunction?
7. Given a file containing sensitive data and residing in a Linux operating system, would setting the permissions to *rw-rw-rw-* cause a potential security issue? If so, which portions of the CIA triad might be affected?
8. Which type of access control would be used in the case where we wish to prevent users from logging in to their accounts after business hours?
9. Explain how the confused deputy problem can allow privilege escalation to take place.
10. What are some of the differences between access control lists and capabilities?

Bibliography

[1] N. Hardy, The confused deputy: (or why capabilities might have been invented), ACM SIGOPS Oper. Syst. Rev. 22 (4) (1988). ISSN: 0163-5980.

[2] K.J. Higgins, CSRF vulnerability: a 'sleeping giant.' Dark Reading. <www.darkreading.com/security/application-security/208804131/index.html>, October 17, 2006 (accessed: December 12, 2010).

[3] F. Callegati, M. Ramilli, Frightened by links, IEEE Secur. Privacy 7 (6) (2009). ISSN: 1540-7993.

[4] L. von Ahn, M. Blum, J. Langford, Telling humans and computers apart automatically, Commun. ACM 47 (2) (2004).

[5] L.J. LaPadula, D.E. Bell, Secure Computer Systems: Mathematical Foundations, vol. l, Mitre Corporation, 1973.

[6] K.J. Biba, Integrity Considerations for Secure Computer Systems, Mitre Corporation, 1975.

[7] T.Y. Lin, Chinese wall security policy—an aggressive model, Fifth Annual Computer Security Applications Conference, 1989.

[8] Maryland Department of Transportation State Highway Administration, Jersey barrier in place on southbound I-95/I-495. Maryland Department of Transportation State Highway Administration, <http://apps.roads.maryland.gov/WebProjectLifeCycle/ProjectPhotos.asp?projectno=PG5725116#>, 2010.