Chapter 14
Controlling and Monitoring Access

THE CISSP TOPICS COVERED IN THIS CHAPTER INCLUDE:

- ✓ **Domain 3.0: Security Architecture and Engineering**
  - 3.7 Understand methods of cryptanalytic attacks
    - 3.7.11 Pass the hash
    - 3.7.12 Kerberos exploitation
- ✓ **Domain 5.0: Identity and Access Management (IAM)**
  - 5.4 Implement and manage authorization mechanisms
    - 5.4.1 Role-based access control (RBAC)
    - 5.4.2 Rule based access control
    - 5.4.3 Mandatory access control (MAC)
    - 5.4.4 Discretionary access control (DAC)
    - 5.4.5 Attribute-based access control (ABAC)
    - 5.4.6 Risk based access control
    - 5.4.7 Access policy enforcement (e.g., policy decision point, policy enforcement point)
  - 5.5 Manage the identity and access provisioning lifecycle
    - 5.5.4 Privilege escalation (e.g., use of sudo, auditing its use)
  - 5.6 Implement authentication systems

Chapter 13, "Managing Identity and Authentication," presented several important topics related to the Identity and Access Management (IAM) domain. This chapter builds on those topics and includes key information on common access control models. It also provides information on how to prevent or mitigate access

control attacks. Be sure to read and study the materials from each of these chapters to ensure complete coverage of this domain's essential material.

Comparing Access Control Models

Chapter 13 focused heavily on identification and authentication. After authenticating subjects, the next step is authorization. The method of authorizing subjects to access objects varies depending on the IT system's access control method.

A *subject* is an active entity that accesses a passive object, and an *object* is a passive entity that provides information to active subjects. For example, when a user accesses a file, the user is the subject and the file is the object.

Comparing Permissions, Rights, and Privileges

When studying access control topics, you'll often come across the terms *permissions*, *rights*, and *privileges*. Some people use these terms interchangeably, but they don't always mean the same thing.

Permissions In general, permissions refer to the access granted for an object and determine what you can do with it. You can grant a user permission to read, write, delete, and/or execute a file on a file server. If you have read permission for a file, you'll be able to open it and read it. You may be granted read and execute permissions for an application file, which gives you permission to run the application. Additionally, you may be granted permissions within a database, allowing you to retrieve or update information in the database.

Rights A right primarily refers to the ability to take an action on an object. For example, a user might have the right to modify

the system time on a computer or the right to restore backed-up data. This is a subtle distinction and not always stressed.

Privileges A privilege is a combination of elevated rights and permissions. For example, an administrator for a computer will have full privileges, granting the administrator full rights and permissions on the computer. The administrator will be able to perform any actions and access any data on the computer.

Understanding Authorization Mechanisms

Access control models use many different types of authorization mechanisms, or methods to control who can access specific objects. Here's a brief introduction to some common mechanisms and concepts:

Implicit Deny A fundamental principle of access control is *implicit deny,* and most authorization mechanisms use it. The implicit deny principle ensures that access to an object is denied unless access has been explicitly granted to a subject. For example, imagine an administrator explicitly grants Jeff Full Control permissions to a file but does not explicitly grant permissions to anyone else. Mary doesn't have any access even though the administrator didn't explicitly deny her access. Instead, the implicit deny principle denies access to Mary and everyone else except for Jeff. You can also think of this as deny by default.

Access Control Matrix Chapter 8, "Principles of Security Models, Design, and Capabilities," covers access control lists and access control matrixes in more detail. In short, an *access control matrix* is a centralized table that includes subjects, objects, and assigned permissions, rights, and privileges. When a subject attempts an action, the system checks the access control matrix to determine if the subject has the appropriate privileges to perform the action. For example, an access control matrix can include a set of files as the objects and a set of users as the subjects. It will show the exact permissions authorized for each user for each file. Note that this covers much more than a single *access control list (ACL)*. In this example, each file listed within

the matrix has a separate ACL that lists the authorized users and their assigned permissions.

Capability List *Capability lists* are a decentralized, distributed method of identifying permissions, rights, and privileges assigned to subjects using tokens or keys. They are different from ACLs in that a capability list is focused on individual subjects (a user or a process). A capability for a user would list the user and its various access permissions, rights, and privileges to individual objects. In contrast, ACLs are focused on individual objects. An ACL for a file would list all the users and/or groups that have authorized access to the file and the specific access granted to each.

The difference between an ACL and a capability list is the focus. ACLs are object-focused and identify access granted to subjects for any specific object. Capability lists are subject-focused and identify the objects that subjects can access.

Constrained Interface Applications use *constrained interfaces* or restricted interfaces to restrict what users can do or see based on their privileges. Users with full privileges have access to all the capabilities of the application. Users with restricted privileges have limited access. Applications constrain the interface using different methods. A common method is to hide the capability if the user doesn't have permission to use it. For example, commands might be available to administrators via a menu or by right-clicking an item, but if a regular user doesn't have permissions, the command does not appear. Other times, the application displays the menu item but shows it dimmed or disabled. A regular user can see the menu item but will not be able to use it. The Clark–Wilson model (covered in Chapter 8) discusses the technical details of how it implements a constrained interface.

Content-Dependent Control *Content-dependent access controls* restrict access to data based on the content within an

object. A database view is a content-dependent control. A view dynamically retrieves specific columns from one or more tables, creating a virtual table. For example, a customer table in a database could include customer names, email addresses, phone numbers, and credit card data. A customer-based view might show only the customer names and email addresses and nothing else. Users granted access to the view can see the customer names and email addresses but cannot access data in the underlying table.

Context-Dependent Control *Context-dependent access controls* require specific activity before granting users access. As an example, consider the data flow for a transaction selling digital products online. Users add products to a shopping cart and begin the checkout process. The first page in the checkout flow shows the products in the shopping cart, the next page collects credit card data, and the last page confirms the purchase and provides instructions for downloading the digital products. The system denies access to the download page if users don't go through the purchase process first. It's also possible to use date and time controls as context-dependent controls. For example, it's possible to restrict access to computers and applications based on the current day and/or time within a time zone. If users attempt to access the resource outside the allowed time, the system denies them access.

Need to Know This principle ensures that subjects are granted access only to what they *need to know* for their work tasks and job functions. Subjects may have clearance to access classified or restricted data but are not granted authorization to the data unless they actually need it to perform a job.

Least Privilege The *principle of least privilege* ensures that subjects are granted only the privileges they need to perform their work tasks and job functions. This is sometimes lumped together with need to know. The only difference is that least privilege will also include rights to take action on a system.

Separation of Duties and Responsibilities The *separation of duties and responsibilities* principle ensures that sensitive functions are split into tasks performed by two or more

employees. It helps prevent fraud and errors by creating a system of checks and balances.

Chapter 16, "Managing Security Operations," covers several related access control topics in more depth. These include need to know, least privilege, and separation of duties.

Defining Requirements with a Security Policy

A *security policy* is a document that defines the security requirements for an organization. It identifies assets that need protection and the extent to which security solutions should go to protect them. Some organizations create a security policy as a single document, and other organizations create multiple security policies, with each one focused on a separate area.

Policies are an important element of access control because they help personnel within the organization understand what security requirements are important. Senior leadership approves the security policy and, in doing so, provides a broad overview of an organization's security needs. However, a security policy usually does not go into details about how to fulfill the security needs or how to implement the policy. For example, it may state the need to implement and enforce separation of duties and least privilege principles but not state how to do so. Professionals within the organization use the security policies as a guide to implement security requirements.

Chapter 1, "Security Governance Through Principles and Policies," covers security policies in more depth. It includes detailed information on standards, procedures, and guidelines.

Introducing Access Control Models

The following sections describe several access control models that you should understand. As an introduction, these access control models are summarized in the following list. The first item in the list introduces a discretionary access control and the rest of the items on the list are nondiscretionary access controls:

Discretionary Access Control A key characteristic of the discretionary access control (DAC) model is that every object has an owner and the owner can grant or deny access to any other subjects. For example, if you create a file, you are the owner and can grant permissions to any other user to access the file. The New Technology File System (NTFS), used on Microsoft Windows operating systems, uses the DAC model.

Role-Based Access Control A key characteristic of the role-based access control (RBAC) model is the use of roles or groups. Instead of assigning permissions directly to users, user accounts are placed in roles and administrators assign privileges to the roles. These roles are typically identified by job functions. If a user account is assigned a role, the user is granted all the privileges assigned to that role. Microsoft Windows operating systems implement RBAC with the use of groups.

Rule-Based Access Control A key characteristic of the rule-based access control model is that it applies predefined global rules to all subjects. As an example, a firewall uses rules that allow or block traffic to all users equally. Rules within the rule-based access control model are sometimes referred to as *restrictions* or *filters*.

Attribute-Based Access Control A key characteristic of the attribute-based access control (ABAC) model is its use of rules that can include multiple attributes. This allows it to be much more flexible than a rule-based access control model that applies the rules to all subjects equally. Many software-defined networks (SDNs) use the ABAC model. Additionally, ABAC allows administrators to create rules within a policy using plain language statements such as “Allow Managers to access the WAN using a mobile device.”

Mandatory Access Control A key characteristic of the mandatory access control (MAC) model is the use of labels applied to both subjects and objects. For example, if a user has a label of top secret, the user can be granted access to a top secret document. In this example, both the subject and the object have matching labels. When documented in a table, the MAC model sometimes resembles a lattice (such as one used for a climbing rosebush), so it is referred to as a lattice-based model.

Risk-Based Access Control A risk-based access control model grants access after evaluating risk. It evaluates the environment and the situation and makes dynamic risk-based decisions using policies embedded within software code. It uses machine learning to make predictive conclusions about current activity based on past activity.

Discretionary Access Control

A system that employs *discretionary access controls* allows the owner, creator, or data custodian of an object to control and define access to that object. All objects have owners, and access control is based on the discretion or decision of the owner. For example, if a user creates a new spreadsheet file, that user is both the creator of the file and the owner of the file. As the owner, the user can modify the permissions of the file to grant or deny access to other users. Data owners can also delegate day-to-day tasks for handling data to data custodians, giving data custodians the ability to modify permissions. Identity-based access control is a subset of DAC because systems identify users based on their identity and assign resource ownership to identities.

A DAC model is implemented using access control lists (ACLs) on objects. Each ACL defines the types of access granted or denied to subjects. It does not offer a centrally controlled management system because owners can alter the ACLs on their objects at will. Access to objects is easy to change, especially when compared to the static nature of mandatory access controls.

Microsoft Windows systems use the DAC model to manage files. Each file and folder has an ACL (also known as a DACL)

identifying the permissions granted to any user or group, and the owner can modify permissions.

Within the DAC model, every object has an owner (or data custodian), and owners have full control over the objects they own. Permissions (such as read and modify for files) are maintained in an ACL, and owners can easily change permissions. This makes the model very flexible.

Nondiscretionary Access Controls

The major difference between discretionary and *nondiscretionary access controls* is in how they are controlled and managed. Administrators centrally administer nondiscretionary access controls and can make changes that affect the entire environment. In contrast, DAC models allow owners to make their own changes, and their changes don't affect other parts of the environment.

In a nondiscretionary access control model, access does not focus on user identity. Instead, a static set of rules governing the whole environment manages access. Non-DAC systems are centrally controlled and easier to manage (although less flexible) and audit. In general, any model that isn't a discretionary access control model is a nondiscretionary model.

Role-Based Access Control

Systems that employ *role-based access control* (RBAC) define a subject's ability to access an object based on the subject's job role. Administrators often implement *role-based access control* using groups or roles.

As an example, a bank may have loan officers, tellers, and managers. Administrators can create a group named Loan Officers, place the user accounts of each loan officer into this group, and then assign appropriate privileges to the group, as shown in Figure 14.1. If the organization hires a new loan officer,

administrators simply add the new loan officer's account into the Loan Officers group, and the new employee automatically has all the same permissions as other loan officers in this group. Administrators would take similar steps for tellers and managers.

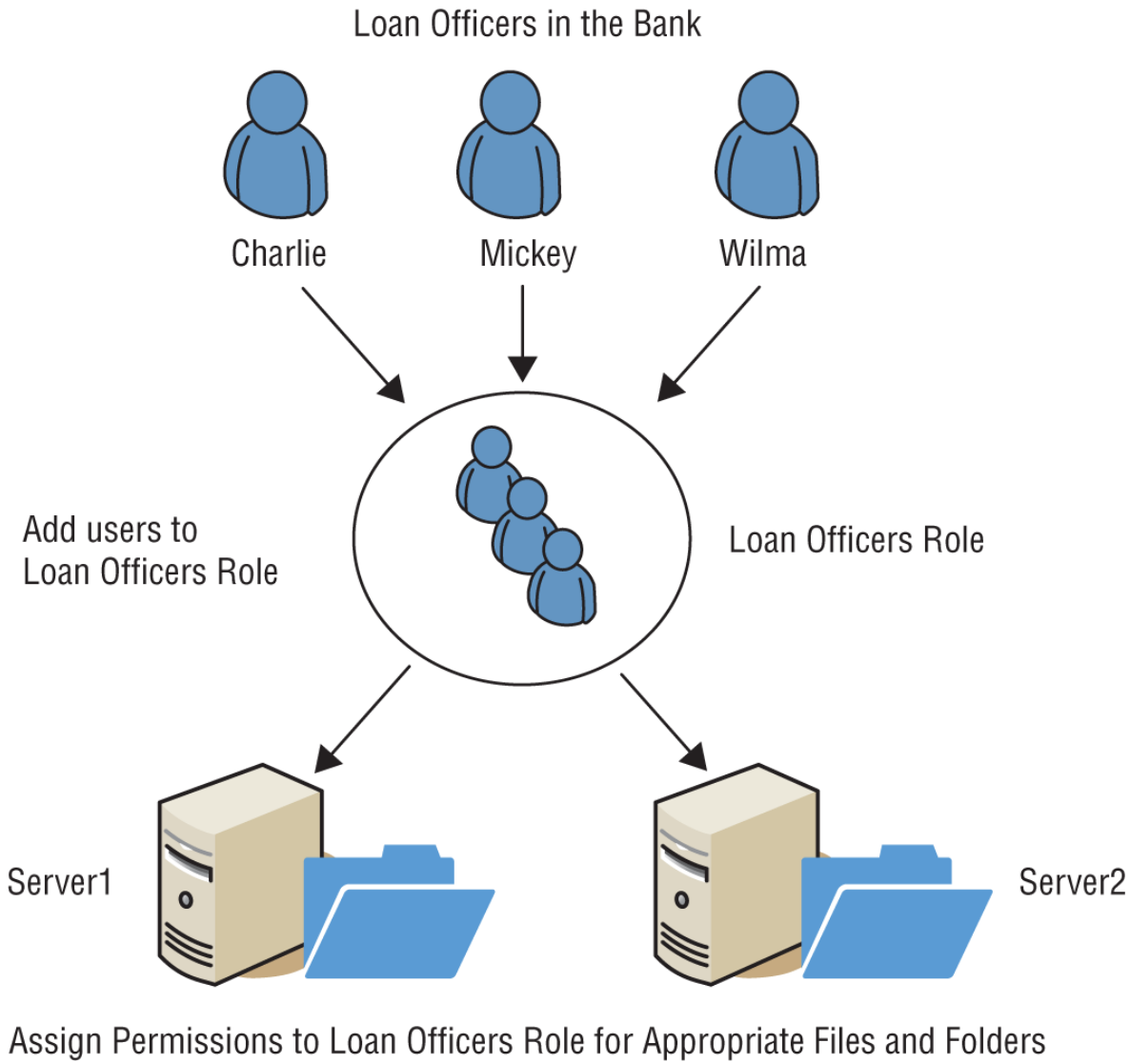

FIGURE 14.1 Role-based access control

This approach helps enforce the principle of least privilege by preventing privilege creep. *Privilege creep* is the tendency for users to accrue privileges over time as their roles and access needs change. Ideally, administrators revoke user privileges when users change jobs within an organization. However, when privileges are assigned to users directly, it is challenging to identify and revoke all of a user's unneeded privileges.

Administrators can easily revoke unneeded privileges by simply removing the user's account from a group. As soon as an administrator removes a user from a group, the user no longer has the privileges assigned to the group. As an example, if a loan officer moves to another department, administrators can simply remove the loan officer's account from the Loan Officers group. This immediately removes all the Loan Officers group privileges from the user's account.

Administrators identify roles (and groups) by job descriptions or work functions. In many cases, this follows the organization's hierarchy documented in an organizational chart. Users who occupy management positions will have greater access to resources than users in a temporary job.

RBAC is useful in dynamic environments with frequent personnel changes because administrators can easily grant multiple permissions simply by adding a new user into the appropriate role. It's worth noting that users can belong to multiple roles or groups. For example, using the same bank scenario, managers might belong to the Managers role, the Loan Officers role, and the Tellers role. This allows managers access to all of the same resources that their employees can access.

Microsoft operating systems implement RBAC with the use of groups. Some groups, such as the local Administrators group, are predefined. However, administrators can create additional groups to match the job functions or roles used in an organization.

A distinguishing point about the RBAC model is that subjects have access to resources through their membership in roles or groups. Roles are based on jobs or tasks, and administrators assign privileges to the role. The RBAC model is useful for enforcing the principle of least privilege because privileges can easily be revoked by removing user accounts from a role.

It's easy to confuse DAC and RBAC because they can both use groups to organize users into manageable units, but they differ in their deployment and use. In the DAC model, objects have owners and owners determine who has access. In the RBAC model, administrators determine subject privileges and assign appropriate privileges to roles or groups. In a strict RBAC model, administrators do not assign privileges to users directly but only grant privileges by adding user accounts to roles or groups.

Another access control model related to RBAC is *task-based access control (TBAC)*. TBAC is similar to RBAC, but instead of being assigned to one or more roles, each user is assigned an array of tasks. These items all relate to assigned work tasks for the person associated with a user account. Under TBAC, the focus is on controlling access by assigned tasks rather than by user identity or job roles.

As an example, Microsoft Project uses TBAC. Each project has multiple tasks. The project manager assigns tasks to project team personnel. Team personnel can address their own tasks (adding comments, indicating progress, and so on), but they cannot address other tasks. Microsoft Project handles the underlying details.

Application Roles

Many applications use the RBAC model because the roles reduce the overall labor cost of maintaining the application. As a simple example, WordPress is a popular web-based application used for blogging and as a content management system.

WordPress includes six roles organized in a hierarchy. The roles are Subscriber, Contributor, Author, Editor, Administrator, and Super Admin. The Subscriber has the fewest privileges, and the Super Admin has the most. Each higher-level role includes all the privileges of the lower-level role(s).

Subscribers can modify some elements of the look and feel of the pages within their user profiles. Contributors can create, edit, and delete their own unpublished posts. Authors can create, edit, and publish posts. They can also edit and delete their own published posts and upload files. Editors can create, edit, and delete any posts. They can also manage website pages, including editing and deleting pages. Administrators can do anything and everything on the site, including managing underlying themes, plug-ins, and users.

Rule-Based Access Control

A *rule-based access control* model uses a set of rules, restrictions, or filters to determine what can and cannot occur on a system. It includes granting a subject access to an object, or granting the subject the ability to perform an action. A distinctive characteristic about rule based access control models is that they have global rules that apply to all subjects.

You may see role-based access control and rule-based access control both abbreviated as RBAC in some other documents. However, the CISSP Content Outline lists them as role-based access control (RBAC) and rule-based access control.

One common example of a rule-based access control model is a firewall. Firewalls include a set of rules or filters within an ACL, defined by an administrator. The firewall examines all the traffic going through it and only allows traffic that meets one of the rules.

Firewalls include a final rule (referred to as the implicit deny rule), denying or blocking all other traffic. The initial rules identify traffic that the firewall will allow. The implicit deny rule denies all other traffic. As an example, the last rule might be `deny all` to indicate the firewall should block all traffic in or out of the network that wasn't previously allowed by another rule.

In other words, if traffic doesn't meet the condition of any previous explicitly defined rule that granted access, then the final rule ensures that the traffic is blocked. This final rule is sometimes viewable in the ACL so that you can see it. Other times, the implicit deny rule is implied as the final rule but is not explicitly stated in the ACL.

Attribute-Based Access Control

Traditional rule-based access control models include global rules that apply to all subjects (such as users) equally. However, an advanced implementation of a rule-based access control is an *attribute-based access control (ABAC)* model. ABAC models use policies that include multiple attributes for rules.

Attributes can be almost any characteristic of users, the network, and devices on the network. For example, user attributes can include group membership, the department where they work, and devices they use such as desktop PCs or mobile devices. The

network can be the local internal network, a wireless network, an intranet, or a wide area network (WAN). Devices can include firewalls, proxy servers, web servers, database servers, and more.

Many software-defined networking (SDN) applications use ABAC models. Chapter 11, "Secure Network Architecture and Components," discusses SDN in greater depth. In short, an SDN separates the infrastructure layer (sometimes called the infrastructure plane or data plane) from the control layer (sometimes called the control plane). This separation gives an organization more freedom to purchase hardware from different sources. The ABAC model provides the organization with more flexibility when managing the SDN.

As an example, a software-defined wide area network (SD-WAN) solution could implement policies to allow or block traffic. Administrators create ABAC policies using plain language statements such as "Allow Managers to access the WAN using tablets or smartphones." This allows users in the Managers role to access the WAN using tablet devices or smartphones. Notice how this improves the rule-based access control model. The rule-based access control applies to all users, but the ABAC can be much more specific.

Mobile device management (MDM) systems, discussed in Chapter 9, "Security Vulnerabilities, Threats, and Countermeasures," can use attributes to identify mobile devices. Chapter 13 gave some attribute examples such as somewhere you are, somewhere you aren't, and context-aware authentication. Context-aware attributes can include the time of day, the type of device, and much more. An MDM system can use these as authentication attributes. For example, imagine an organization wants to grant users access to the network during work hours and only when using a specific Android-based phone. The MDM system can verify these attributes and allow the user to log on when the attributes match.

Mandatory Access Controls

A *mandatory access control (MAC)* model relies on the use of classification labels, discussed in Chapter 5, "Protecting Security

of Assets." Each classification label represents a security *domain*, or a realm of security. A security domain is a collection of subjects and objects that share a common security policy. For example, a security domain could have the label Secret, and the MAC model would protect all objects with the Secret label in the same manner. Subjects are only able to access objects with the Secret label when they have a matching Secret label that indicates they are cleared to access Secret information. Note that users may have more than one label if they are cleared to access multiple levels of information. Additionally, the requirement for subjects to gain the Secret label is the same for all subjects.

Users have labels assigned to them based on their clearance level, which is a form of privilege. Similarly, objects have labels, which indicate their level of classification or sensitivity. For example, the U.S. military uses the labels Top Secret, Secret, Confidential, and Unclassified to classify data. Administrators can grant access to Top Secret data to users with Top Secret clearances. However, administrators cannot grant access to Top Secret data to users with lower-level clearances such as Secret and Confidential.

Organizations in the private sector often use labels such as confidential (or proprietary), private, sensitive, and public. Governments use labels mandated by law, but private sector organizations are free to use whatever labels they choose.

The MAC model is often referred to as a lattice-based model. Figure 14.2 shows an example of a lattice-based MAC model. It is reminiscent of a lattice in a garden, such as a rose lattice used to train climbing roses. The horizontal lines labeled Confidential, Private, Sensitive, and Public mark the upper bounds of the classification levels. For example, the area between Public and Sensitive includes objects labeled Sensitive (the upper boundary). Users with the Sensitive label can access Sensitive data.

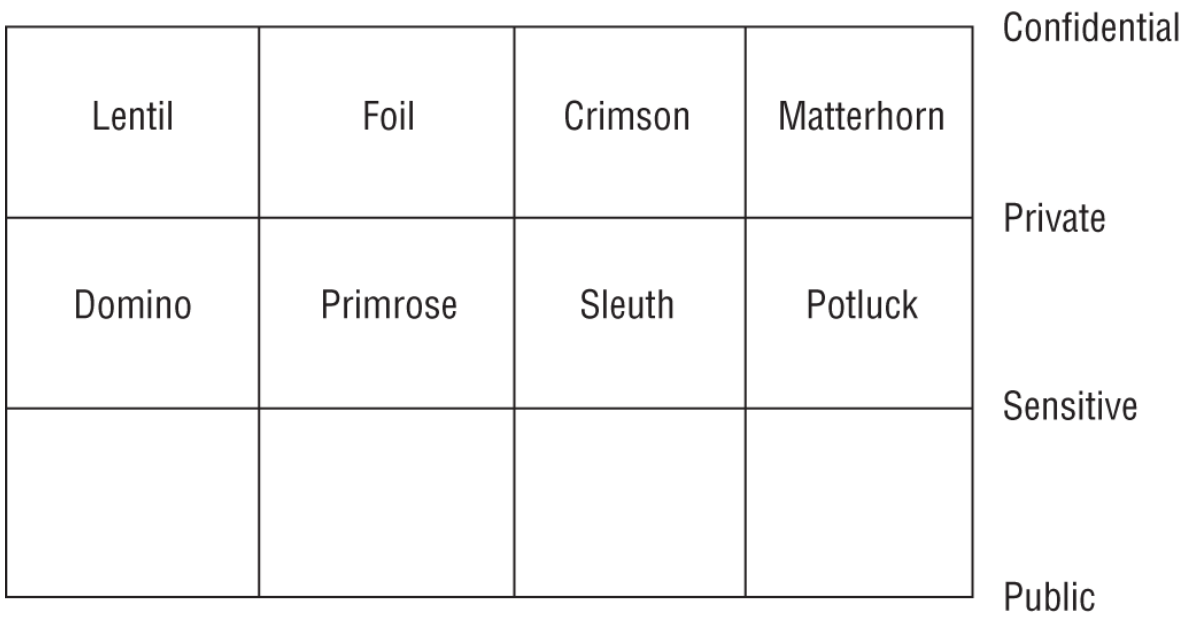

FIGURE 14.2 A representation of the boundaries provided by lattice-based access controls

The MAC model also allows labels to identify more defined security domains. Within the Confidential section (between Private and Confidential), there are four separate security domains labeled Lentil, Foil, Crimson, and Matterhorn. These all include Confidential data but are maintained in separate compartments for an added layer of protection. Users with the Confidential label also require the additional label to access data within these compartments. For example, to access Lentil data, users need to have both the Confidential label and the Lentil label.

Similarly, the compartments labeled Domino, Primrose, Sleuth, and Potluck include Private data. Users need the Private label and one of the labels in this compartment to access the data within that compartment.

The labels in Figure 14.2 are names of World War II military operations, but an organization can use any names for the labels. The key is that these sections provide an added level of compartmentalization for objects such as data. Notice that Sensitive data (between the Public and Sensitive boundaries) doesn't have any additional labels. Users with the Sensitive label can be granted access to any data with the Sensitive label.

Personnel within the organization identify the labels and define their meanings as well as the requirements to obtain the labels. Administrators then assign the labels to subjects and objects. With the labels in place, the system determines access based on the assigned labels.

Using compartmentalization with the MAC model enforces the *need to know* principle. Users with the Confidential label are not automatically granted access to compartments within the Confidential section. However, if their job requires them to have access to certain data, such as data with the Crimson label, an administrator can assign them the Crimson label to grant them access to this compartment.

The MAC model is prohibitive rather than permissive, and it uses an implicit deny philosophy. If users are not specifically granted access to data, the system denies them access to the associated data. The MAC model is more secure than the DAC model, but it isn't as flexible or scalable.

Security classifications indicate a hierarchy of sensitivity. For example, if you consider the military security labels of Top Secret, Secret, Confidential, and Unclassified, the Top Secret label includes the most sensitive data and unclassified is the least sensitive. Because of this hierarchy, someone cleared for Top Secret data is cleared for Secret and less sensitive data. However, classifications don't have to include lower levels. It is possible to use MAC labels so that a clearance for a higher-level label does not include clearance for a lower-level label.

A key point about the MAC model is that every object and every subject has one or more labels. These labels are predefined, and the system determines access based on assigned labels.

Classifications within a MAC model use one of the following three types of environment:

Hierarchical Environment *A hierarchical environment* relates various classification labels in an ordered structure from low security to medium security to high security, such as Confidential, Secret, and Top Secret, respectively. Each level or classification label in the structure is related. Clearance in one level grants the subject access to objects in that level as well as to all objects in lower levels but prohibits access to all objects in higher levels. For example, someone with a Top Secret clearance can access Top Secret data and Secret data.

Compartmentalized Environment In a *compartmentalized environment*, there is no relationship between one security domain and another. Each domain represents a separate isolated compartment. To gain access to an object, the subject must have specific clearance for the object's security domain.

Hybrid Environment A *hybrid environment* combines both hierarchical and compartmentalized concepts so that each hierarchical level may contain numerous subdivisions that are isolated from the rest of the security domain. A subject must have the correct clearance and the need to know data within a specific compartment to gain access to the compartmentalized object. A hybrid MAC environment provides granular control over access but becomes increasingly difficult to manage as it grows. Figure 14.2 is an example of a hybrid environment.

Risk-Based Access Control

Risk-based access control is relatively new, and the implementation can be quite complex. The model attempts to evaluate risk by considering several different elements, such as:

- The environment
- The situation
- Security policies

In this context, a security policy is software code that makes risk-based decisions based on available data. An organization would modify the choices within the software to support their needs.

For example, consider an information system containing patient information and used by medical professionals. Doctors, nurses, and others working in the emergency room (ER) of a hospital need access to this data for any patient who shows up in the ER. In this scenario, the environment is the ER, and the situation is a medical emergency. Security policies will likely consider this a low risk and grant full access to patient data to doctors and nurses.

Consider the same database that is used by personnel in the pharmacy department. In this case, the environment is the pharmacy, and the situation is the dispensing of medication. Security policies will likely consider this to be medium or low risk. The risk-based model would grant some access to the patient data to identify any potential adverse drug interactions. However, the model would prevent access to the full medical history of patients.

These are simplified examples of an environment. Within cybersecurity, the environment can include items such as the location using the IP address. Some low-risk IP addresses may be internal IP addresses and Internet-based IP addresses of users who have previously signed in. High-risk IP addresses could be from foreign countries, anonymized IP addresses, users signed in from two or more IPs in different countries, and users signed in from unfamiliar locations.

The situation may include what a device is doing. As an example, most Internet of Things (IoT) devices have predictable behavior. If an IoT device suddenly starts flooding a network with malicious traffic, the risk based model could determine the device is now a high risk and block its access to the network.

Two other things can be checked or required before the policy grants access:

Multifactor Authentication The system will deny access to users logging on with just one factor of authentication.

Compliant Mobile Devices The policy may require that smartphones and tablets meet specific security requirements, such as an up-to-date operating system and device encryption.

A risk-based access control model can sometimes use binary rules to control access. For example, either a user logged in using multifactor authentication or they didn't. However, other policies may require the model to implement machine learning capabilities. It would then make predictive conclusions about current activity based on past actions and grant or block access based on these conclusions.

A risk-based access control model that examines mobile devices for compliance may interact with an existing mobile device management (MDM) system. Chapter 9 covers mobile device management in more depth.

Implementing Authentication Systems

Authentication systems simplify the management of authentication on the Internet and in internal networks. Chapter 13 discusses federated identity management (FIM) and single sign-on (SSO) concepts in more depth, but as a reminder, FIM allows different organizations to use federations for SSO. For example, after an employee logs on to Company A's network, they can then access resources on Company B's network without logging on again.

Implementing SSO on the Internet

Beyond federated identity management systems, many sites support SSO to simplify the user experience. They also provide security to users by ensuring their credentials on one site are not shared with other sites.

Imagine you want to transfer money from Bank A to Bank B. You could give your Bank A credentials to Bank B and have them transfer the money. Sound scary? You bet. You should never be required to give your credentials to any third party. Solutions such as SAML, OAuth, OpenID, and OIDC help solve this

problem. They share authentication, authorization, or profile information about a user, and some solutions share all three.

XML

Extensible Markup Language (XML) goes beyond describing how to display the data by actually describing the data. XML can include tags to describe data as anything desired. For example, the following tag identifies the data as the results of taking an exam: `<ExamResults>Passed</ExamResults>`.

Databases from multiple vendors can import and export data to and from an XML format, making XML a common language used to exchange information. Many specific schemas exist, and if companies agree on what schema to use, they can easily share information. Many cloud-based providers use XML-based languages to share information for authentication and authorization. They don't use XML as it is but instead use other languages based on XML.

SAML

Security Assertion Markup Language (SAML) is an open XML-based standard commonly used to exchange authentication and authorization (AA) information between federated organizations. It provides SSO capabilities for browser access.

The Organization for the Advancement of Structured Information Standards (OASIS Open), a nonprofit consortium that encourages open standards development, adopted SAML 2.0 as a standard in 2005 and has maintained it since then. SAML 2.0 is a convergence of SAML 1.1, the Liberty Alliance Identity Federation Framework (ID-FF) 1.2, and Internet2's Shibboleth 1.3.

The SAML 2.0 specification utilizes three entities: the principal (or user), the service provider (SP), and the identity provider (IdP). For example, imagine Sally is accessing her investment account at `ucanbeamillionaire.com`. The site requires her to log on to access her account, and the site uses SAML 2.0.

Principal or User Agent For simplicity, think of Sally as the principal. She's trying to access her investment account at

ucanbeamillionaire.com.

Service Provider (SP) or Relying Party In this scenario, the ucanbeamillionaire.com site is providing the service and is the service provider.

Identity Provider (IdP) or Asserting Party This is a third party that holds the user authentication and authorization information.

When Sally accesses the service provider site, she identifies herself to the SP. The SP then determines the relevant identity provider and redirects Sally to the IdP where she enters her credentials. After she completes the authentication process, the IdP responds to the SP with XML messages (SAML assertions) validating or rejecting Sally's credentials. Upon a successful authentication, the IdP provides Sally's session attributes and what she is authorized to access to the SP. The SP then grants Sally access to her account.

The IdP can send three types of XML messages during assertions. The following are the statements that may be included in a SAML assertion:

Authentication Statements An authentication statement provides proof that the user agent provided the proper credentials, identifies the identification method, and identifies the time the user agent logged on.

Attribute Statements An attribute statement can be any information about the user agent including their entitlements.

Authorization Statements An authorization statement indicates whether the user agent is authorized to access the requested service. If the message indicates access is denied, it indicates why.

Many cloud service providers include SAML in their solutions because it simplifies the services for their customers. SAML provides authentication, attribute, and authorization statements in its assertions.

SAML is a popular SSO standard on the Internet. It is used to exchange authentication and authorization (AA) information.

OAuth

OAuth 2.0 (implying open authorization) is an authorization framework described in RFC 6749 and maintained by the Internet Engineering Task Force (IETF). Many companies on the internet use it to share account information with third-party websites.

For example, imagine you have a social media platform account, and you download an app called Acme that can interact with your social media account and schedule posts in advance. When you try to use the feature in the Acme app, it redirects you to the social media site. That site prompts you to log on, shows you what permissions the Acme app will access, and then asks if you want to authorize the Acme app to access your social media account. If you approve, the social media platform sends the Acme app an authorization token. The app may accept and enter the authorization token directly, or you may need to enter it into the app's settings. When the app accesses the social media account, it sends an API message and includes the token. Note that this doesn't provide authentication. Instead, it authorizes access to the account. A primary benefit is that you never provide your social media account credentials to the Acme app. Even if the Acme app is compromised, it does not expose your credentials.

Many online sites support OAuth 2.0 but not OAuth 1.0, and OAuth 2.0 is not backward compatible with OAuth 1.0.

OAuth is an authorization framework, not an authentication protocol. It exchanges API messages and uses a token to show that access is authorized.

OpenID Connect

OpenID Connect (OIDC) is an authentication layer using the OAuth 2.0 authorization framework. A key point is that it provides both authentication and authorization. OIDC is maintained by the OpenID Foundation.

OIDC uses a JavaScript Object Notation (JSON) Web Token (JWT), also called an ID token. OpenID Connect uses a web service to retrieve the JWT. In addition to providing authentication, the JWT can also include profile information about the user.

Most of this occurs behind the scenes, but you can see it in action by logging onto eBay with a Google account. These processes and interfaces change over time, but the general steps are as follows:

1. If you don't have a Google account, create one first.
2. Ensure you're logged out of eBay and Google, go to http://ebay.com, and click Sign In.
3. Click Continue With Google. A dialog box opens, prompting you to enter your Google email. It also indicates what Google will share with http://ebay.com.
4. Enter your email address and press Enter.
5. Enter your password and click Next.
6. If you've enabled 2-Step Verification on your Google account, you'll be prompted to get the code and enter it.

You don't need to complete the creation of an eBay account with your Google account. However, if you choose to do so, click the Create Account button. You'll now be logged on to eBay using

your Google account. If you log out of eBay and try to log on again, all you need to do is click Sign In and then click Continue with Google. As long as you're still logged on with Google, you'll be logged into eBay without any more steps.

OAuth and OIDC are used with many web-based applications to share information without sharing credentials. OAuth provides authorization. OIDC uses the OAuth framework for authorization and builds on the OpenID technologies for authentication. OIDC uses JSON Web Tokens.

Comparing SAML, OAuth, and OpenID Connect

It's easy to mix up the differences between SAML, OAuth, and OIDC. This section summarizes key points of each one and points out some of the differences.

The following bullets outline the key points about SAML:

- SAML 2.0 is an open XML-based standard.
- OASIS adopted it as a standard in 2005.
- It utilizes three entities: a principal (such as a user), a service provider (such as a website), and an identity provider (a third party that holds the authentication and authorization information).
- It can provide authentication, authorization, and attribute information on the principal.

The following bullets outline the key points about OAuth:

- It's an authorization framework, not an authentication protocol.
- RFC 6749 describes OAuth 2.0.
- It exchanges information using APIs.

- An app obtains an access token from an identity provider.
- Later, the app includes the access token for authorization.

The following bullets outline the key points about OpenID Connect (OIDC):

- OIDC is an authentication layer using OAuth 2.0.
- It provides both authentication and authorization.
- It builds on OpenID (a deprecated standard) but uses a JSON Web Token.

Implementing SSO on Internal Networks

SSO solutions are also used on internal networks. Kerberos is the most common. Network access methods allow users to access internal networks from remote locations (such as at home). Two common remote access protocols are RADIUS and TACACS+. In addition to supporting SSO, RADIUS and TACACS+ provide authentication, authorization, and accounting.

AAA Protocols

Several protocols provide authentication, authorization, and accounting and are referred to as AAA protocols. These provide centralized access control with remote access systems such as virtual private networks (VPNs) and other types of network access servers (NASs). They help protect internal LAN authentication systems and other servers from remote attacks. If you are using a separate system for remote access, a successful attack on the system only affects the remote access users. In other words, the attacker won't have access to internal accounts.

These AAA protocols use the access control elements of identification, authentication, authorization, and accounting as described in Chapter 13. They ensure that a user has valid credentials to authenticate and verify that the user is authorized to connect to the remote access server based on the user's proven identity. Additionally, the accounting element can track the user's

network resource usage, which can be used for billing purposes. Some common AAA protocols are covered next.

Kerberos

Ticket authentication is a mechanism that employs a third-party entity to prove identification and provide authentication. The most common and well-known ticket-based authentication system is *Kerberos*. The primary purpose of Kerberos is authentication. After users authenticate and prove their identity, Kerberos uses their proven identity to issue tickets, and user accounts present these tickets when accessing resources.

The Kerberos name is borrowed from Greek mythology. A three-headed dog named Kerberos, sometimes referred to as Cerberus, guards the gates to the underworld. The dog faces inward, preventing escape rather than denying entrance.

Kerberos offers a single sign-on solution for users and protects logon credentials. Kerberos version 5 relies on symmetric-key cryptography (also known as secret-key cryptography) using the Advanced Encryption Standard (AES) symmetric encryption protocol. Kerberos provides confidentiality and integrity for authentication traffic using end-to-end security and helps protect against eavesdropping and replay attacks. Chapter 6, "Cryptography and Symmetric Key Algorithms," covers symmetric key encryption in greater depth.

Many of the Kerberos roles are on a single server, but they can be installed on different servers. Larger networks sometimes separate them to increase performance, but smaller networks typically have one Kerberos server performing all of the different roles.

Kerberos uses several different elements that are important to understand:

Key Distribution Center The Key Distribution Center (KDC) is the trusted third party that provides authentication services. Kerberos uses symmetric-key cryptography to authenticate clients to servers. All clients and servers are registered with the KDC, and it maintains the secret keys for all network members.

Kerberos Authentication Server The authentication server hosts the functions of the KDC: a ticket-granting service (TGS) and an authentication service (AS). However, it is possible to host the ticket-granting service on another server. The *authentication service* verifies or rejects the authenticity and timeliness of tickets. This server is often called the KDC.

Ticket A ticket is an encrypted message that provides proof that a subject is authorized to access an object. It is sometimes called a service ticket (ST). Subjects (such as users) request tickets to access objects (such as files), and if they have authenticated and are authorized to access the object, the KDC issues them a ticket. Kerberos tickets have specific lifetimes and usage parameters. Once a ticket expires, a client must request a renewal or a new ticket to continue communications with any server.

Ticket-Granting Ticket A ticket-granting ticket (TGT) provides proof that a subject has authenticated through a KDC and is authorized to request tickets to access other objects. A TGT is encrypted and includes a symmetric key, an expiration time, and the user's IP address. Subjects present the TGT when requesting tickets to access objects.

Kerberos Principal The KDC issues tickets to Kerberos principals. A Kerberos principal is typically a user but can be any entity that can request a ticket.

Kerberos Realm Generically, a realm is an area controlled or ruled by something. A Kerberos realm is a logical area (such as a domain or network) ruled by Kerberos. Principals within the realm can request tickets from the Kerberos KDC, and the KDC can issue tickets to principals in the realm.

Kerberos requires a database of accounts, typically stored in a directory service such as Microsoft's Active Directory (AD). It

exchanges tickets between clients, network servers, and the KDC to prove identity and provide mutual authentication. This allows a client to request resources from the server, with both the client and server having assurances of the identity of the other. These encrypted tickets also ensure that login credentials, session keys, and authentication messages are never transmitted in cleartext.

The Kerberos login process works as follows:

1. The user types a username and password into the client.
2. The client generates a request, including the plaintext username and domain of the user (but not the password), and sends the request to the Kerberos authentication server.
3. The authentication server verifies the username against its database of known users.
4. The KDC generates a session key that will be used by the client and the Kerberos server. It encrypts this with a hash of the user's password. The KDC also generates an encrypted timestamped TGT.
5. The KDC then transmits the encrypted session key and the encrypted timestamped TGT to the client.
6. The client installs the TGT for use until it expires. The client also decrypts the session key using a hash of the user's password.

Note that the user's password is never transmitted over the network, but it is verified. The server encrypts a symmetric key using a hash of the user's password, and it can only be decrypted with a hash of the user's password. As long as the user enters the correct password, this step works. However, it fails if the user enters the incorrect password.

When a client wants to access an object, such as a resource hosted on the network, it must request a ticket through the Kerberos server. The following steps are involved in this process:

1. The client sends its TGT back to the KDC with a request for access to the resource.
2. The KDC verifies that the TGT is valid and checks its access control matrix to verify that the user has sufficient privileges to access the requested resource.
3. The TGS generates a service ticket and sends it to the client.
4. The client sends the service ticket to the server or service hosting the resource.
5. The server or service hosting the resource verifies the validity of the service ticket with the KDC.
6. Once identity and authorization are verified, Kerberos activity is complete. The server or service host then opens a session with the client and begins communications or data transmission.

Kerberos is a versatile authentication mechanism that works over local LANs, remote access, and client/server resource requests. However, Kerberos presents a single point of failure—the KDC. If the KDC is compromised, the secret key for every system on the network is also compromised. Also, if a KDC goes offline, no subject authentication can occur.

It also has strict time requirements, and the default configuration requires that all systems be time-synchronized within 5 minutes of each other. If a system is not synchronized or the time is changed, a previously issued TGT will no longer be valid, and the system will not be able to receive any new tickets. In effect, the client will be denied access to any protected network resources.

Administrators often configure a time synchronization system within a network. In an Active Directory domain, one domain controller (DC) synchronizes its time with an external Network Time Protocol (NTP) server. All other DCs synchronize their time

with the first DC. All other systems synchronize their time with one of the DCs when they log on. Kerberos uses port 88.

RADIUS

Remote Authentication Dial-in User Service (RADIUS) centralizes authentication for remote access connections, such as with VPNs or dial-up access. It is typically used when an organization has more than one network access server (or remote access server). A user can connect to any network access server, which then passes on the user's credentials to the RADIUS server to verify authentication and authorization and to track accounting. In this context, the network access server is the RADIUS client, and a RADIUS server acts as an authentication server. The RADIUS server also provides AAA services for multiple remote access servers.

Many internet service providers (ISPs) use RADIUS for authentication. Users can access the ISP from anywhere, and the ISP server then forwards the user's connection request to the RADIUS server.

Organizations can also use RADIUS, and organizations often implement it with location-based security. For example, if the user connects with an IP address, the system can use geolocation technologies to identify the user's location. Although it isn't as common today, some users still have Integrated Services Digital Network (ISDN) dial-up lines and use them to connect to VPNs. The RADIUS server can use callback security for an extra layer of protection. Users call in, and after authentication, the RADIUS server terminates the connection and initiates a call back to the user's predefined phone number. If a user's authentication credentials are compromised, the callback security prevents an attacker from using them.

RADIUS uses the User Datagram Protocol (UDP) by default and encrypts only the password's exchange. It doesn't encrypt the entire session, but RADIUS can use other protocols to encrypt the data session. The current version is defined in RFC 2865. RFC 6614, designated as Experimental, defines how RADIUS can use

Transport Layer Security (TLS) over Transmission Control Protocol (TCP).

When using TLS, RADIUS uses TCP port 2083. RADIUS uses UDP port 1812 for RADIUS authentication and authorization messages and UDP port 1813 for RADIUS accounting messages.

RADIUS provides AAA services between network access servers and a shared authentication server. The network access server is the client of the RADIUS authentication server.

TACACS+

Cisco developed Terminal Access Controller Access Control System Plus (TACACS+) and later released it as an open standard. It provides several improvements over the earlier versions and over RADIUS.

It separates authentication, authorization, and accounting into separate processes, which can be hosted on three different servers if desired. Additionally, TACACS+ encrypts all of the authentication information, not just the password, as RADIUS does. TACACS+ uses TCP port 49, providing a higher level of reliability for the packet transmissions.

Zero-Trust Access Policy Enforcement

Organizations are increasingly designing their networks and infrastructure using *zero-trust* principles. Unlike traditional "moat and castle" or defense-in-depth designs, zero-trust presumes that there is no trust boundary and no network edge. Instead, each action is validated when requested as part of a continuous authentication process, and access is only allowed after policies are checked, including elements like identity, permissions, system configuration and security status, threat intelligence data review, and security posture.

Figure 14.3 shows NIST's logical diagram of a zero-trust architecture (ZTA). Note that a *subject*'s use of a system (which is untrusted) connects through a *policy enforcement point*, allowing trusted transactions to the enterprise resources. The *policy engine* makes policy decisions based on rules that are then acted on by the *policy administrator*.

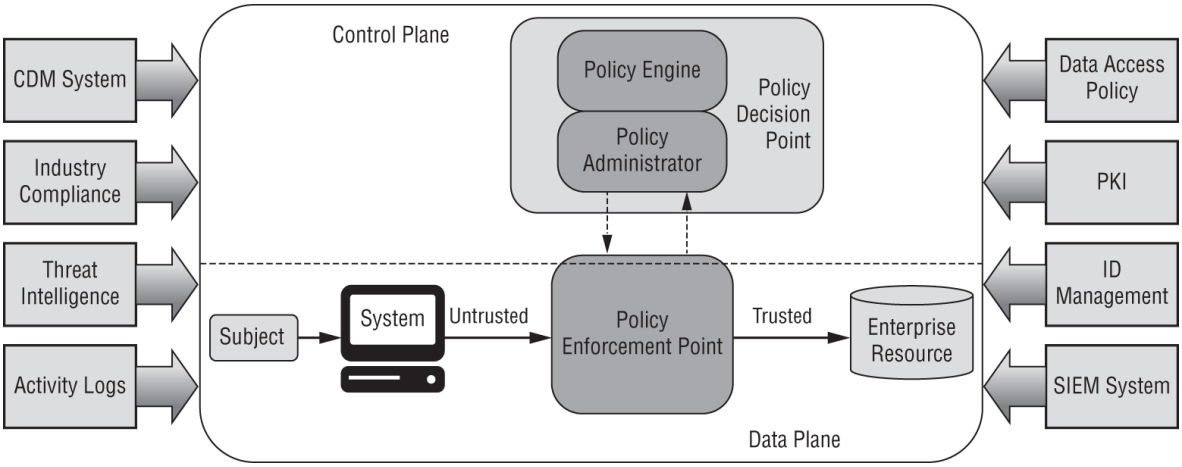

FIGURE 14.3 NIST Zero-Trust core trust logical components

Here are the key zero-trust components that you should be familiar with:

- *Subjects* are the users, services, or systems that request access or attempt to use rights.
- *Policy engines* make policy decisions based on both rules and external systems like those shown above: threat intelligence, identity management, and SIEM devices, to name just a few. They use a trust algorithm that makes the decision to grant, deny, or revoke access to a given resource based on the factors used for input to the algorithm. Once a decision is made, it is logged and the policy administrator takes action based on the decision.
- *Policy administrators* are not individuals. Rather they are components that establish or remove the communication path between subjects and resources, including creating session-specific authentication tokens or credentials as needed. In cases where access is denied, the policy

administrator tells the policy enforcement point to end the session or connection.

- Together, the policy engine and policy administrator are known as the *policy decision point*.
- *Policy enforcement points* communicate with policy administrators to forward requests from subjects and to receive instructions from them about connections to allow or end. While the policy enforcement point is shown as a single logical element above, they are commonly deployed with a local client or application and a gateway element that is part of the network path to services and resources.

You can read the NIST publication about zero-trust at https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf.

Understanding Access Control Attacks

As mentioned in Chapter 13, one of the goals of access control is to prevent unauthorized access to objects. This includes access to any information system, including networks, services, communications links, and computers, and unauthorized access to data. In addition to controlling access, IT security methods seek to prevent unauthorized disclosure of data and unauthorized alteration of assets and to provide consistent availability of resources. In other words, IT security methods attempt to prevent the loss of confidentiality, loss of integrity, and loss of availability.

Security professionals need to be aware of common attack methods so that they can take proactive steps to prevent them, recognize them when they occur, and respond appropriately. The following sections provide a quick review of risk elements and cover common access control attacks.

While this section focuses on access control attacks, it's important to realize that there are many other types of attacks covered in other chapters. For example, Chapter 6 covers various cryptanalytic attacks.

Crackers, Hackers, and Attackers

Crackers are malicious individuals who are intent on waging an attack against a person or system. They attempt to crack the security of a system to exploit it, and they are typically motivated by greed, power, or recognition. Their actions can result in loss of property (such as data and intellectual property), disabled systems, compromised security, negative public opinion, loss of market share, reduced profitability, and lost productivity. In many situations, crackers are simply criminals.

In the 1970s and 1980s, hackers were defined as technology enthusiasts with no malicious intent. However, the media now uses the term *hacker* in place of *cracker*. Its use is so widespread that the definition has changed.

To avoid confusion, in this book we typically use the term *attacker* for malicious intruders. An attack is any attempt to exploit the vulnerability of a system and compromise confidentiality, integrity, and/or availability.

Risk Elements

Chapter 2, "Personnel Security and Risk Management Concepts," covers risk and risk management in more depth, but it's worth reiterating some terms in the context of access control attacks. A *risk* is the possibility or likelihood that a threat will exploit a vulnerability, resulting in a loss such as harm to an asset. A *threat* is a potential occurrence that can result in an undesirable outcome. This includes potential attacks by criminals or other attackers. It also includes natural occurrences such as floods or earthquakes, as well as accidental acts by employees. A

vulnerability is any type of weakness. The weakness can be due to a flaw or limitation in hardware or software. It can also be the absence of a security control, such as the absence of antivirus software on a computer.

Risk management attempts to reduce or eliminate vulnerabilities or reduce the impact of potential threats by implementing controls or countermeasures. It is not possible, or financially desirable, to eliminate risk. Instead, an organization focuses on reducing the risks that can cause it the most harm.

Common Access Control Attacks

Access control attacks attempt to bypass or circumvent access control methods. As mentioned in Chapter 13, access control starts with identification, authentication, and authorization, and access control attacks often try to steal user credentials. After attackers have stolen a user's credentials, they can launch an online *impersonation* attack by logging in as the user and accessing the user's resources. In other cases, an access control attack can bypass authentication mechanisms and just steal the data.

This book covers multiple attacks, and the following sections cover common attacks directly related to access control.

Privilege Escalation

Privilege escalation refers to any situation that gives users more privileges than they should have. Normally, a regular user would have enough privileges to perform their job but no more. This includes rights and permissions on their own computer and on network servers, such as file servers.

Chapter 13 covers most of the topics in objective 5.5, “Manage the identity and access provisioning life cycle.” However, we chose to place privilege escalation in this chapter because it is a key element in many successful attacks.

In contrast, local administrators have full rights and permissions on local computers, and domain administrators have full rights and permissions within a domain. Regular users should not have the same privileges as administrators.

Attackers use privilege escalation techniques to gain elevated privileges. As an example, imagine a regular user opens a malicious attachment in a phishing email. The malware gives the attacker the same privileges as the user, which are severely limited in most situations.

Privilege escalation is often described as horizontal privilege escalation and vertical privilege escalation. Attackers combine the two to compromise as many systems and accounts as they can within a network.

Horizontal is side to side, and vertical is up and down. If you have trouble remembering the difference between the two, think about watching a sunset (or sunrise) over the ocean. The horizon is the theoretical line going from left to right, separating the sky from the earth.

Imagine an attacker gains control of a regular user's account, such as after a successful phishing attack. Horizontal privilege escalation gives an attacker similar privileges as the first compromised user, but from other accounts.

Vertical privilege escalation provides an attacker with significantly greater privileges. After compromising a regular user's account, an attacker can use vertical privilege escalation techniques to gain administrator privileges on the user's computer. The attacker can then use horizontal privilege escalation techniques to access other computers in the network. This horizontal privilege escalation throughout the network is also known as lateral movement. The attacker can then attempt vertical escalation techniques on every other compromised computer.

The “Mimikatz” section, later in this chapter, explains how attackers can use this tool to gain more and more privileges within a network. After infecting a regular user's computer, attackers use Mimikatz to gain administrator privileges on the user's computer and then move throughout the network, gaining more privileges. Given enough time, the attacker will often gain domain administrator privileges.

Chapter 13 discussed service accounts within the context of service authentication. These are frequently called *managed service accounts* because administrators create them to run services or applications and manage them. As an example, it's common to set the password so that it never expires but manually change the password regularly.

An important consideration with managed service accounts is to ensure they have only the privileges needed by the service or application. For example, imagine you install a database application. The application needs to run under the context of a service account with specific rights and permissions. The easiest way to do this is to use the LocalSystem account because it has full administrative privileges on the local system, and you don't have to manage the password. However, the easiest way is not the correct way. Instead, you would create a new account and give it only the needed rights and permissions.

Using the *su* and *sudo* Commands

Linux systems have a root user account, sometimes called a superuser account. The root account on Linux is similar to an administrator account on Windows systems. Users can log on to the root account with root as the username and the root password. However, doing so isn't normally recommended, because it's easy to forget that you're logged on as a superuser.

Instead, administrators log on with a regular account when doing daily tasks. When they need to run commands as the root account, they use the `su` command (short for switch user or substitute user). The `su` command switches to the root account by default and prompts the user to enter the root account password.

After running commands with elevated permissions, administrators can return to their regular accounts.

Another alternative is the `sudo` command, sometimes referred to as *superuser do*. Administrators with root privileges can grant permission to any user to run the `sudo` command by adding them to the sudo group. This is similar to adding a user to the Administrators group on Windows systems. When users are added to the sudo group, they don't need the password to the root account but instead use their own credentials. Once logged in, the user can prefix commands with `sudo` to run the command as root. Logs will record any commands using `sudo` with the user's account, providing auditing capabilities. In contrast, if the user switches to the su account with the `su` command, logs will record the activity using the su account, not the user's account.

Real World Scenario

Privilege Escalation with PowerShell

Imagine an application is installed on a Windows server using the LocalSystem account instead of a service account. Later, an attacker discovers and exploits a vulnerability in the application, giving the attacker access to the LocalSystem account with full local administrative privileges. Many Windows systems have PowerShell installed by default, so the attacker can now use it as fileless malware and run PowerShell scripts as an administrator.

The attacker can start with some network reconnaissance. As an example, the `Get-ADComputer` cmdlet will retrieve a listing of all computers in an Active Directory domain. The attacker can then run PowerShell scripts on any remote computer.

By default, the execution policy for PowerShell is set to Restricted, indicating you can't run PowerShell scripts. For example, the execution policy causes the following command to fail:

```
powershell.exe .\hello.ps1
```

The `hello.ps1` script simply displays Hello World to the screen. Instead of calling the script, you can use the `Get-Content` cmdlet to read the script, and then pass the text to PowerShell with the `Invoke-Expression` cmdlet.

```
powershell.exe "& {Get-Content .\hello.ps1 | Invoke-
Expression}
```

The key here is that using the LocalSystem account provides full administrative access to the local system. Whenever possible, it's best to create a service account instead of using the LocalSystem account.

Password Attacks

Passwords are the weakest form of authentication, and there are many types of password attacks. If an attacker is successful in a password attack, the attacker can access the account and access resources authorized to the account. If an attacker discovers a root or administrator password, the attacker can access any other account and its resources. If attackers discover passwords for privileged accounts in a high-security environment, the environment's security can never be fully trusted again. The attacker could have created other accounts or backdoors to access the system. Instead of accepting the risk, an organization may choose to rebuild the entire system from scratch.

A *strong password* is sufficiently long, uses a combination of character types, and helps prevent password attacks. The phrase "sufficiently long" is a moving target and dependent on the usage and the environment. Chapter 13 discusses password policies, strong passwords, and the use of passphrases. The important point is that longer passwords are stronger than shorter passwords when using the same character types, and longer passwords with multiple character types create even stronger passwords.

Although security professionals usually know what makes a strong password, many users do not, and it is common for users to create short passwords with only a single character type. Past data breaches help illustrate this. After the data breach, attackers often post stolen databases with account names and hashed passwords. Analysis of these databases shows that many users still use simple passwords such as 12345, 123456, 1234567, 12345678, 123456789, password, and abc123.

Organizations rarely store passwords in cleartext. Instead, they use a strong hashing function such as SHA-3 and create a hash of the password. They then store the hash instead of the password. Chapter 6 covers hashing in more depth. As a reminder, a hash is simply an alphanumeric string created by executing a hashing algorithm against a string of characters or file. A hashing algorithm will always produce the same hash when run against the same password.

When a user authenticates, the system hashes the provided password and typically sends the hash to an authentication server in an encrypted format. The authentication server decrypts the message containing the hash and then compares that decrypted value to the stored hash for the user. If the hashes match, the system authenticates the user.

It's important to use strong hashing functions when hashing passwords. Many password attacks succeed when organizations have used weak hashing functions, such as Message Digest 5 (MD5). MD5 is compromised and not recommended for use as a cryptographic hashing function. It should not be used to hash passwords.

It's also important to change default passwords. IT professionals know this for computers, but this knowledge hasn't extended consistently to IoT devices and embedded systems. Chapter 9 covers IoT devices and embedded systems in more depth. If the default password isn't changed, anyone who knows the default password can log in and cause problems.

The following sections describe common password attacks using a variety of methods. Some of these attacks are possible against online accounts. As an example, an attacker could try to guess the usernames and passwords on an online web server or web application. In other attacks, an attacker steals an account database and then cracks the passwords using an offline attack. Account databases can be customer databases, or operating system files such as the Windows-based Security Account Manager (SAM) file or the `/etc/shadow` file on Linux systems.

Dictionary Attack

A *dictionary attack* is an attempt to discover passwords by using every possible password in a predefined database or list of common or expected passwords. In other words, an attacker starts with a database of words commonly found in a dictionary. Dictionary attack databases also include character combinations widely used as weak passwords but not found in dictionaries. For example, you will probably see passwords such as 123456 and password in password-cracking dictionaries.

Additionally, dictionary attacks often scan for one-upped-constructed passwords. A one-upped-constructed password is a previously used password, but with one character different. For example, password1 is one-upped from password, as are password2, 1password, and passXword. Attackers often use this approach when generating rainbow tables (discussed later in this chapter).

Some people think that using a foreign word as a password will beat dictionary attacks. However, password-cracking dictionaries can, and often do, include foreign words.

Brute-Force Attack

A *brute-force attack* is an attempt to discover passwords for user accounts by systematically attempting all possible combinations of letters, numbers, and symbols. Attackers don't typically type these in manually but instead have programs that can programmatically try all the combinations.

A *hybrid attack* attempts a dictionary attack and then performs a type of brute-force attack with one-upped-constructed passwords.

Longer and more complex passwords take more time and are costlier to crack than simple passwords. As the number of possibilities increases, the cost of performing an exhaustive attack goes up. In other words, the longer the password and the more character types it includes, the more secure it is against brute-force attacks.

Passwords and usernames are typically stored in an account database file on secured systems. However, instead of being stored as plaintext, systems and applications commonly hash passwords and store only the hash values.

The following three steps illustrate one way that a user might authenticate with a hashed password:

1. The user enters credentials such as a username and password.
2. The user's system hashes the password and sends the hash to the authenticating system.
3. The authenticating system compares this hash to the hash stored in the password database file. If it matches, it indicates the user entered the correct password.

This approach provides two protections. Passwords do not traverse the network in cleartext, which would make them susceptible to sniffing attacks. Password databases do not store passwords in cleartext, but instead store them as hashes. Passwords stored as cleartext would be much easier for attackers to read if they gained access to the password database.

However, password attacker tools look for a password that creates the same hash value as an entry stored in the account database file. If they're successful, they can use the password to log on to the account. As an example, imagine the password IPassed has a stored hash value of 1A5C7G hexadecimal (though the actual hash would be much longer). A brute-force password tool would take these steps:

1. Guess a password.
2. Calculate the hash of the guessed password.
3. Compare the calculated hash against the stored hash in the offline database.
4. Repeat steps 1 through 3 until a guessed password has the same hash as a stored password.

This is also known as comparative analysis or reverse-hash matching. When the password-cracking tool finds a matching hash value, it indicates that the guessed password is very likely the original password. The attacker can now use this password to impersonate the user.

If two separate passwords create the same hash, it results in a collision. Collisions aren't desirable, and better hashing functions

are collision resistant. Unfortunately, some hashing functions (such as MD5) allow an attacker to create a different password that results in the same hash as a hashed password stored in the account database file. This is one of the reasons that MD5 is not recommended for hashing passwords today.

With the speed of modern computers and the ability to employ distributed computing, brute-force attacks prove successful against even some strong passwords. The actual time it takes to discover passwords depends on the algorithm used to hash them and the power of the computer.

Many attackers are using GPUs in brute-force attacks. In general, GPUs have more processing power than most CPUs in desktop computers. Additionally, it's relatively easy for a do-it-yourselfer to create a multiple-GPU computer and use it to crack passwords in offline databases.

However, longer passwords take longer to crack than shorter and simple passwords. For example, a 15-character password using uppercase and lowercase characters takes longer to crack than an 8-character password. Similarly, a complex 15-character password using all four character types (uppercase, lowercase, numbers, and special characters) takes longer to crack than a 15-character password using only uppercase and lowercase characters.

With enough time, attackers can discover any hashed password using an offline brute-force attack. However, longer passwords result in sufficiently longer times, making it infeasible for attackers to crack them.

Spraying Attack

A *spraying attack* is a special type of brute-force attack. Attackers use spraying attacks in online password attacks, attempting to bypass account lockout security controls.

Usually, a system will lock out an account if the same user enters the wrong password too many times within a short amount of

time, such as 30 minutes. In a spraying attack, a program uses the same guessed password but loops through a list of different accounts and different systems. When it finishes the list, it picks another password and loops through the list again. The list is long, and it typically takes the program as long as 15 to 30 minutes to loop through it.

Imagine the lockout policy locks out an account if the same account tries the wrong password five times within 30 minutes and the spraying attack loops through the list in 15 minutes. After entering the incorrect password twice (30 minutes), the 30-minute timer resets. The account will not be locked out.

Credential Stuffing Attack

Credential stuffing is sometimes confused with password spraying, but the two attacks are different. Password spraying attempts to bypass account lockout policies, whereas credential stuffing only checks a single username and password on each site.

Imagine that Gus has hundreds of accounts on various sites such as eBay, Netflix, and Disney+. He's become overwhelmed with tracking all of these credentials, so he uses the same credentials on every site. Later, one of these sites is the victim of an attack. Malicious actors download the credential database and discover all of the usernames and passwords in an offline attack, including Gus's credentials. They then use an automated tool to try Gus's credentials on hundreds of sites (or more).

If people use different passwords on all sites, a credential stuffing attack will fail. However, many people continue to use the same credentials on multiple sites.

Birthday Attack

A *birthday attack* focuses on finding collisions. Its name comes from a statistical phenomenon known as the *birthday paradox*. The birthday paradox states that if there are 23 people in a room, there is a 50 percent chance that any two of them will have the

same birthday—not the same year but the same month and day, such as March 30.

With February 29 in a leap year, there are only 366 possible days in a year. With 367 people in a room, you have a 100 percent chance of getting at least two people with the same birthdays. Reduce this to only 23 people in the room, and you still have a 50 percent chance that any two have the same birthday.

This is similar to finding any two passwords with the same hash. Imagine a simple, hypothetical, hashing function that could only create 366 different hashes. In that case, an attacker with a sample of only 23 hashes has a 50 percent chance of discovering two passwords that create the same hash. Hashing algorithms can create many more than 366 different hashes, but the point is that the birthday attack method doesn't need all possible hashes to see a match.

From another perspective, imagine that you are one of the people in the room and you want to find someone else with the same birthday as you. In this example, you'll need 23 people in the room to reach the same 50 percent probability of finding someone else with the same birthday.

Similarly, it is possible for some tools to come up with another password that creates the same hash of a given hash. For example, if you know that the hash of the administrator account password is 1A5C7G, some tools can identify a password that will create the same hash of 1A5C7G. It isn't necessarily the same password, but if it can create the same hash, it is just as effective as the original password.

You can reduce the success of birthday attacks by using hashing algorithms with enough bits to make collisions computationally infeasible and use salts (discussed in the “Rainbow Table Attack” section next). There was a time when security experts considered MD5 (using 128 bits) to be strong enough to protect passwords. However, computing power continues to improve, and MD5 is no longer recommended as a cryptographic hash. SHA-3 (short for Secure Hash Algorithm version 3) can use as many as 512 bits and is more collision resistant to brute-force attacks than MD5.

Computing power continues to improve, so at some point, SHA-3 will be replaced with another hashing algorithm with longer hashes and/or stronger cryptology methods used to create the hash.

Rainbow Table Attack

It takes a long time to find a password by guessing it, hashing it, and then comparing it with a valid password hash. However, a *rainbow table* reduces this time by using large databases of precomputed hashes. Attackers create rainbow tables by:

1. Guessing a password
2. Hashing the guessed password
3. Putting both the guessed password and the hash of the guessed password into the rainbow table

A password cracker can then compare every hash in the rainbow table against the hash in a stolen password database file. A traditional password-cracking tool must guess the password and hash it before it can compare the hashes, which takes time. However, when using the rainbow table, the password cracker doesn't spend any time guessing and calculating hashes. It simply compares the hashes until it finds a match. This can significantly reduce the time it takes to crack a password.

Many different rainbow tables are available for free download, but they are large. For example, an MD5-based rainbow table using all four character types for an eight-character password is about 460 gigabytes in size. Instead of downloading these tables, many attackers create their own using tools such as `rtgen` (available in Kali Linux) and scripts freely available on the Internet.

Many systems commonly salt passwords to reduce the effectiveness of rainbow table attacks. A *salt* is a group of random

bits added to a password before hashing it. Cryptographic methods add the additional bits before hashing it, making it significantly more difficult for an attacker to use rainbow tables against the passwords. *Argon2*, *bcrypt,* and *Password-Based Key Derivation Function 2 (PBKDF2)* are some algorithms used to salt passwords.

However, given enough time, attackers can still crack salted passwords using a brute-force attack. Adding a pepper to a salted password increases the security, making it more difficult to crack. *Salts* are random numbers stored in the same database holding the hashed passwords, so if an attacker gets the database, the attacker also has the salts for the passwords. A *pepper* is a large constant number stored elsewhere, such as a configuration value on a server or a constant stored within application code.

The practice of salting passwords was specifically introduced to thwart rainbow table attacks, but it also thwarts the effectiveness of offline dictionary and brute-force attacks. These offline attacks must calculate the hash of the guessed passwords, and if the stored passwords include salts, the attacks fail unless they also discover the salt. Again, the use of a pepper stored outside the database holding the salted, hashed passwords makes all of these attacks even more difficult.

Mimikatz

Benjamin Delpy released Mimikatz in 2011 to perform some experiments in Windows security while learning C. It has since become a popular tool used by attackers and penetration testers alike. Several exploitation frameworks, such as Metasploit, include Mimikatz, and it is still maintained and updated on GitHub, a software development platform hosting open source projects.

You may be wondering why we're discussing a tool created in 2011. The reason is simple—it continues to work. Part of the reason Mimikatz continues to work is that developers continue to update it.

Chapter 13 discusses single sign-on (SSO) capabilities in depth. In short, SSO lets users sign on once and access other network resources without signing on again. However, SSO methods store credentials in memory, and Mimikatz exploits this by reading memory credentials.

Here are some capabilities of Mimikatz:

Read Passwords from Memory Plaintext passwords and PINs stored in the Local Security Authority Subsystem Service (LSASS) process can be extracted and read. For example, the `sekurlsa::logonpasswords` command will display the user ID and password for users currently logged on to the system. It's also possible to obtain the password hashes.

Extract Kerberos Tickets Mimikatz includes a Kerberos module that can access the Kerberos API. The upcoming "Kerberos Exploitation Attack" section discusses several ticket-based attacks that are possible using Mimikatz and similar tools.

Extract Certificates and Private Keys Mimikatz includes a Windows CryptoAPI module. This module can extract certificates on a system as well as the private keys associated with these certificates.

Read LM and NTLM Password Hashes in Memory Although it is possible to prevent Windows systems from storing LM hashes in the local Security Account Manager database, some Windows systems still create the hash and store it in memory.

Read Cleartext Passwords in Local Security Authority Subsystem Service (LSASS) The LSASS doesn't normally store passwords in cleartext, but malware can modify the registry

to enable digest authentication. Once enabled, Mimikatz can read the passwords.

List Running Processes Attackers can use this capability to identify processes that they can use to pivot their attack against other targets.

Attackers can run Mimikatz as fileless malware on remote systems. One way is with a PowerShell script, such as `Invoke-Mimikatz`, that loads Mimikatz in memory without saving the Mimikatz files on disk. Mimikatz can then perform any of its functions on the remote computer.

Although attackers and security professionals may know Mimikatz as a famous and magical tool, it isn't as well known by typical IT professionals. The danger here is that the fixes to block Mimikatz aren't implemented consistently, allowing attackers to use it frequently.

Pass-the-Hash Attack

A pass-the-hash (PtH) attack allows an attacker to send a captured hash of a password to an authenticating service. Normally, the user would enter a password on the client, and the client would then create the password hash and send the hash. In this attack, the attacker doesn't need to know the actual password.

Penetration testers and attackers use Mimikatz and other tools (such as DCSync) to capture hashes, and then use the hashes to simulate the login process. They can enter the user ID and the hash into the tool and send them to an authentication server. PtH attacks are primarily associated with Windows systems using NT LAN Manager (NTLM) or Kerberos, but other systems can also be vulnerable.

After attackers gain access to a single system in a network, they can then launch a PtH attack. The overall steps are as follows:

1. Use a tool such as Mimikatz to capture user hashes. These are stored in the `lsass.exe` process running in memory. The Mimikatz command (entered on one line) is:

```
"privilege::debug" "log passthehash.log"
"sekurlsa::logonpasswords"
```

If anyone with administrator privileges recently logged on, it will capture the administrator's user ID and hash.

2. The attacker then uses the credentials to authenticate. The attacker can log on as the user on the local system or remotely to an authentication server such as a domain controller in a Microsoft Active Directory (AD) domain.
3. Once logged in, the attacker can use the account to move laterally throughout the network. As a simple example, the PsExec tool can execute commands on remote systems. Just opening the command prompt on the remote system gives the attacker the ability to run simple commands to perform more network reconnaissance. Of course, the attacker can repeat these three steps on the remote system.

A popular tool used in step 3 on Microsoft systems is PsExec. PsExec is part of the Sysinternals process utilities (PsTools), a free download offered by Microsoft at http://learn.microsoft.com/en-us/sysinternals. PsTools is a suite of command-line utilities used to connect to remote computers. Administrators use it to access the command prompt on remote systems. They can then run any command prompt commands, list processes, reboot computers, dump event logs, and more.

There are several steps administrators can take to mitigate PtH attacks. However, this is a moving target. Attackers are continually looking at ways to bypass the mitigations, and Microsoft has been providing updates to limit PtH attacks. The best protection is to prevent the infection of the first computer.

If someone is logged on to the first system with administrator privileges, it's game over. The attacker can use those privileges to access any other system in the network. However, even if an

administrator has not logged on to that machine, the attacker can still move laterally through the network. By repeating the steps on every other system on the network, the attacker is sure to find one where an administrator recently logged on.

Kerberos Exploitation Attack

Kerberos was discussed earlier within the context of single sign-on (SSO) in the earlier section "Implementing SSO on Internal Networks." Microsoft's AD uses Kerberos as the primary authentication protocol. Unfortunately, Kerberos is susceptible to several exploitation attacks using open source tools such as Mimikatz.

Other tools often used in Kerberos exploitation attacks are Rubeus and Impacket. Rubeus is an open source tool written in C# and used on Windows systems. Impacket is an open source collection of modules written in Python and used on Linux systems.

Kerberos exploitation attacks include the following:

Overpass the Hash This is an alternative to the PtH attack used when NTLM is disabled on a network. Even if NTLM is disabled on a network, systems still create an NTLM hash and store it in memory. An attacker can request a ticket-granting ticket (TGT) with the user's hash and use this TGT to access network resources. This is sometimes called *pass the key*.

Pass the Ticket In a pass the ticket attack, attackers attempt to harvest tickets held in the `lsass.exe` process. After harvesting the tickets, attackers inject the ticket to impersonate a user.

Silver Ticket A silver ticket uses the captured NTLM hash of a service account to create a ticket-granting service (TGS) ticket. Service accounts (user accounts used by services) use TGS tickets instead of TGT tickets. The silver ticket grants the attacker all the privileges granted to the service account.

Golden Ticket If an attacker obtains the hash of the Kerberos service account (KRBTGT), they can create tickets at will within AD. This gives them so much power it is referred to as having a

golden ticket. The KRBTGT account encrypts and signs all Kerberos tickets within a domain with a hash of its password. Because the password never changes, the hash never changes, so an attacker needs to learn the hash only once. If an attacker gains access to a domain administrator account, they can then log on to a domain controller remotely and run Mimikatz to extract the hash. This allows attackers to create forged Kerberos tickets and request ticket-granting service (TGS) tickets for any service.

Kerberos Brute Force Attackers can use the Python script `kerbrute.py` on Linux systems or Rubeus on Windows systems. In addition to guessing passwords, these tools can guess usernames. Kerberos reports whether or not usernames are valid.

ASREPRoast ASREPRoast identifies users that don't have Kerberos preauthentication enabled. Kerberos preauthentication is a security feature within Kerberos that helps prevent password-guessing attacks. When preauthentication is disabled, attackers can send an authentication request to a KDC. The KDC will reply with a TGT, encrypted with the client's password as the key. The attacker can then perform an offline attack to decrypt the ticket and discover the client's password.

Kerberoasting Kerberoasting collects encrypted TGS tickets. Service accounts (user accounts used by services) use TGS tickets instead of TGT tickets. After harvesting these tickets, attackers can crack them offline.

> A TGS ticket is used by services running in the context of a user account. This attack attempts to find users that don't have Kerberos preauthentication.

Sniffer Attack

Sniffing captures packets sent over a network with the intent of analyzing the packets. A sniffer (also called a packet analyzer or protocol analyzer) is a software application that captures traffic traveling over the network. Administrators use sniffers to analyze network traffic and troubleshoot problems.

Of course, attackers can also use sniffers. A *sniffer attack* (also called a snooping attack or eavesdropping attack) occurs when an

attacker uses a sniffer to capture information transmitted over a network. They can capture and read any data sent over a network in cleartext, including passwords.

Wireshark is a popular protocol analyzer available as a free download. Figure 14.4 shows Wireshark with the contents of a relatively small capture and demonstrates how attackers can capture and read data sent over a network in cleartext.

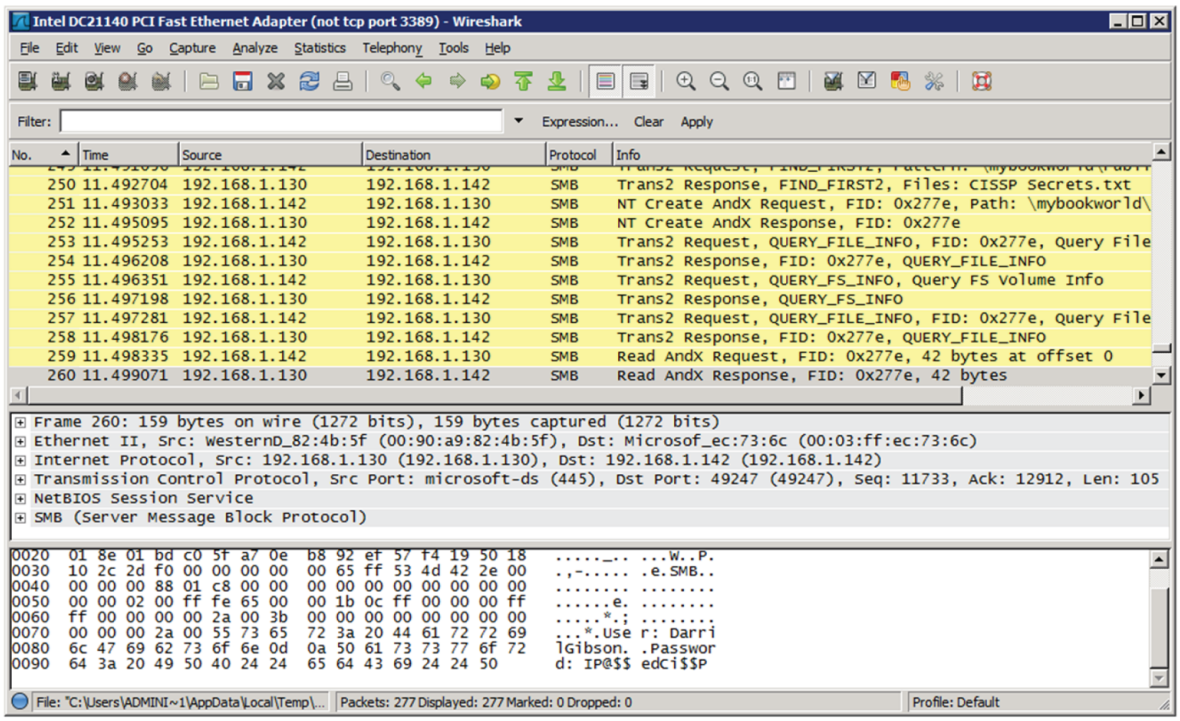

FIGURE 14.4 Wireshark capture

The top pane shows packet 260 selected, and you can see the contents of this packet in the bottom pane. It includes the text `User: DarrilGibson Password: IP@$$edCi$$P`. If you look at the first packet in the top pane (packet number 250), you can see that the name of the opened file is `CISSP Secrets.txt`.

The following techniques can prevent successful sniffing attacks:

- Encrypt all sensitive data (including passwords) sent over a network. Attackers cannot easily read encrypted data with a sniffer. For example, Kerberos encrypts tickets to prevent attacks, and attackers cannot easily read the contents of these tickets with a sniffer.

- Avoid the use of insecure protocols such as HTTP, FTP, and Telnet and use secure protocols such as HTTPS, SFTP, and SSH.
- Use onetime passwords when encryption is not possible or feasible. Onetime passwords prevent the success of sniffing attacks because they are only used once. Even if an attacker captures a onetime password, the attacker is not able to use it.
- Protect network devices with physical security. Controlling physical access to routers and switches prevents attackers from installing sniffers on these devices.
- Monitor the network for signatures from sniffers. Intrusion detection systems can monitor the network for sniffers and will raise an alert when they detect a sniffer on the network.

Spoofing Attacks

Spoofing (also known as masquerading or impersonation) is pretending to be something, or someone, else. There is a wide variety of spoofing attacks. As an example, an attacker can use someone else's credentials to enter a building or access an IT system. Some applications spoof legitimate login screens. One attack brought up a login screen that looked exactly like the operating system logon screen. When the user entered credentials, the fake application captured the user's credentials, and the attacker used them later. Some phishing attacks mimic this with bogus websites.

In an IP spoofing attack, the attacker replaces a valid source IP address with a false one to hide their identity or to impersonate a trusted system. Other types of spoofing used in access control attacks include email spoofing and phone number spoofing:

Email Spoofing Spammers spoof the email address in the From field to make an email appear to come from another source. Phishing attacks often do this to trick users into thinking the email is coming from a trusted source. The Reply To field can be a different email address, and email programs typically don't

display this until a user replies to the email. By this time, they often ignore it or don't notice it.

Phone Number Spoofing Caller ID services allow users to identify the phone number of any caller. Phone number spoofing allows a caller to replace this number with another one, which is a common technique on Voice over Internet Protocol (VoIP) systems. One technique attackers have been using recently is to replace the actual calling number with a phone number that includes the same area code as the called number. This makes it look like it's a local call.

Core Protection Methods

The following list summarizes many security precautions that protect against access control attacks. However, it's important to realize that this isn't a comprehensive list of protections against all types of attacks. You'll find additional controls that help prevent attacks covered throughout this book.

Control physical access to systems. An old saying related to security is that if an attacker has unrestricted physical access to a computer, the attacker owns it. If attackers can gain physical access to an authentication server, they can steal the password file in a very short time. Once they have the password file, they can crack the passwords offline. If attackers successfully download a password file, all passwords should be considered compromised.

Control electronic access to files. Tightly control and monitor electronic access to all important data, including files and customer databases containing passwords. End users and those who are not account administrators have no need to access a password database file for daily work tasks. Security professionals should investigate any unauthorized access to password database files immediately.

Hash and salt passwords. Use protocols such as Argon2, bcrypt, and PBKDF2 to salt passwords and consider using an external pepper to further protect passwords. Combined with the use of strong passwords, salted and peppered passwords are

extremely difficult to crack using rainbow tables or other methods.

Use password masking. Ensure that applications don't display passwords in cleartext by default. Instead, mask the display of the password by displaying an alternate character such as an asterisk (*). This reduces shoulder surfing attempts, but users should be aware that an attacker might be able to learn the password by watching the user type the keys on the keyboard. When a system requires users to enter excessively long passwords, developers should consider an option to show the passwords in cleartext.

Deploy multifactor authentication (MFA). Deploy multifactor authentication, such as using biometrics or token devices. When an organization uses MFA, attackers are not able to access a network if they discover just a password. Many online services, such as Google, now offer multifactor authentication as an additional measure of protection.

Use account lockout controls. Account lockout controls help prevent online password attacks. They lock an account after the incorrect password is entered a predefined number of times. Account lockout controls typically use clipping levels that ignore some user errors but take action after reaching a threshold. For example, it's common to allow a user to enter the incorrect password as many as five times before locking the account. For systems and services that don't support account lockout controls, such as most File Transfer Protocol (FTP) servers, extensive logging along with an intrusion detection system (IDS) can protect the server.

Account lockout controls help prevent an attacker from guessing a password in an online account. However, this does not prevent an attacker from using a password-cracking tool against a stolen database file containing hashed passwords.

Use last logon notification. Many systems display a message including the time, date, and location (such as the computer name or IP address) of the last successful logon. If users pay attention to this message, they might notice if someone else logged on to their account. For example, if a user logged on to an account last Friday but the last logon notification indicates someone accessed the account on Saturday, it indicates a problem. Users who suspect someone else is logging on to their accounts can change their passwords or report the issue to a system administrator. If it occurs with an organizational account, users should report it following the organization's security incident reporting procedures.

Educate users about security. Properly trained users have a better understanding of security and the benefit of using stronger passwords. Inform users that they should never share or write down their passwords. Administrators might write down long, complex passwords for the most sensitive accounts, such as administrator or root accounts, and store these passwords in a vault or safety deposit box. Offer tips to users on how to create strong passwords, such as with password phrases, and how to prevent shoulder surfing. Also, let users know the dangers of using the same password for all online accounts, such as banking accounts and gaming accounts. When a user uses the same passwords for all these accounts, a successful attack on a gaming system can give attackers access to the user's bank accounts. Users should also know about common social engineering tactics.

Summary

This chapter covered several different access control models. With a discretionary access control (DAC) model, all objects have an owner, and the owner has full control over the object. Role-based access control (RBAC) models use roles or groups that often match the hierarchy of an organization. Administrators place users into roles and assign privileges to the roles based on jobs or tasks. Rule-based access controls use global rules that apply to all subjects equally. Attribute-based access control (ABAC) models use policies that include attributes to assign

access. Mandatory access control (MAC) models require all objects to have labels, and access is based on subjects having a matching label. Risk-based access controls evaluate the environment and the situation and make risk-based decisions based on security policies. The emerging zero-trust model presumes no inherent trust and continuously verifies each request against dynamic policies.

Several internet-based authentication systems provide users with single sign-on (SSO) capabilities. SAML is an XML-based standard used to exchange authentication and authorization information. OAuth 2.0 is an authorization framework. OIDC uses OAuth 2.0, and it builds on the technologies used by OpenID. It uses a JSON Web Token as an ID token.

Kerberos is a popular SSO authentication protocol using tickets for authentication in internal networks. It uses a database of subjects, symmetric cryptography, and time synchronization of systems to issue tickets. RADIUS and TACACS+ are common authentication, authorization, and accounting (AAA) protocols.

Access control attacks include privilege escalation techniques to gain more rights and permissions. Passwords are a common authentication mechanism, and several types of attacks attempt to crack passwords. Password attacks include dictionary attacks, brute-force attacks, spraying attacks, credential stuffing attacks, birthday attacks, rainbow table attacks, pass-the-hash attacks, Kerberos exploitation attacks, and sniffer attacks.

Study Essentials

Identify common authorization mechanisms.
Authorization ensures that the requested activity or object access is possible, given the authenticated identity's privileges. For example, it ensures that users with appropriate privileges can access files and other resources. Common authorization mechanisms include implicit deny, access control lists, access control matrices, capability lists, constrained interfaces, content-dependent controls, and context-dependent controls. These

mechanisms enforce security principles such as need to know, the principle of least privilege, and separation of duties.

Describe key concepts of the discretionary access control (DAC) model. With the DAC model, all objects have owners, and the owners can modify permissions. Each object has an access control list defining permissions, such as read and modify for files. All other models are nondiscretionary models, and administrators centrally manage nondiscretionary controls.

Describe key concepts of the role-based access control (RBAC) model. RBAC models use job roles, and users gain privileges when administrators place their accounts into a role or group. Taking a user out of a role removes the permissions granted through the role membership.

Describe key concepts of the rule-based access control model. Rule-based access control models use a set of rules, restrictions, or filters to determine access. A firewall's access control list includes a list of rules that define what access is allowed and what access is blocked.

Describe key concepts of the attribute-based access control (ABAC) model. An ABAC model is an advanced implementation of a rule-based access control model, applying rules based on attributes. Software-defined networks (SDNs) often use an ABAC model.

Describe key concepts of the mandatory access control (MAC) model. The MAC model uses labels to identify security domains. Subjects need matching labels to access objects. The MAC model enforces the need to know principle and supports a hierarchical environment, a compartmentalized environment, or a combination of both in a hybrid environment. It is frequently referred to as a lattice-based model.

Describe key concepts of the risk-based access control model. A risk-based access control model evaluates the environment and the situation and makes decisions based on software-based security policies. It can control access based on multiple factors such as a user's location, determined by IP addresses, whether the user has logged on with multifactor

authentication, and the user's device. Advanced implementations can use machine learning to evaluate risk.

Understand single sign-on methods used on the Internet. SSO is a mechanism that allows subjects to authenticate once and access multiple objects without authenticating again. Security Assertion Markup Language (SAML) is an open XML-based standard used to exchange authentication and authorization information. OAuth 2.0 is an authorization framework described in RFC 6749 and supported by many online sites. OASIS maintains OpenID Connect (OIDC). OIDC provides both authentication and authorization by using the OAuth 2.0 framework and building on the OpenID standard.

Describe Kerberos. Kerberos is the most common SSO method used within organizations. The primary purpose of Kerberos is authentication. It uses symmetric cryptography and tickets to prove identification and provide authentication. One server synchronizes its time with a Network Time Protocol (NTP) server, and all clients within a network synchronize with the same time.

Understand the purpose of AAA protocols. Several protocols provide centralized authentication, authorization, and accounting services. Network access (or remote access) systems use AAA protocols. For example, a network access server is a client to a RADIUS server, and the RADIUS server provides AAA services. RADIUS uses UDP and encrypts the password only. TACACS+ uses TCP and encrypts the entire session.

Describe privilege escalation. Attackers use privilege escalation techniques to gain additional privileges after exploiting a single system. They typically try to gain additional privileges on the exploited systems first. They can also reach other systems in a network and attempt to gain elevated privileges on them. Limiting privileges given to service accounts reduces the success of some privilege escalation attacks.

Explain zero-trust principles. Zero-trust presumes that there is no trust boundary and no network edge. Instead, each action is validated when requested as part of a continuous

authentication process, and access is only allowed after policies are checked. The key components of a zero-trust architecture are the policy engine and policy administrator (which together are known as the policy decision point) and the policy enforcement point.

Know about Kerberos exploitation attacks. Kerberos attacks attempt to exploit weaknesses in Kerberos tickets. In some attacks, they capture tickets held in the `lsass.exe` process and use them in pass the ticket attacks. A silver ticket grants the attacker all the privileges granted to a service account. Attackers can create golden tickets after obtaining the hash of the Kerberos service account (KRBTGT), giving them the ability to create tickets at will within Active Directory.

Know how brute-force and dictionary attacks work. Brute-force and dictionary attacks are carried out against a stolen password database file or the system's logon prompt. They are designed to discover passwords. In brute-force attacks, all possible combinations of keyboard characters are used, whereas a predefined list of possible passwords is used in a dictionary attack. Account lockout controls prevent their effectiveness against online attacks.

Written Lab

1. Describe the primary difference between discretionary and nondiscretionary access control models.
2. List at least three standards used to provide single sign-on (SSO) capabilities on the Internet.
3. Identify the PowerShell cmdlet that allows you to run PowerShell commands indirectly.
4. Name a tool that is commonly used in the pass the hash and Kerberos exploitation attacks for privilege escalation.

Review Questions