

ITPro™
SERIES

Windows
& .NET MAGAZINE

eBooks

Best Practices for Managing Linux and UNIX Servers

By Dustin Puryear

 netiq
Work Smarter.



Table of Contents

Chapter 1 Infrastructure and Data Security	20
The Security Policy	20
Policy Summary	20
Responsible Parties	21
Policy	22
Physical Security	23
System Security	23
Operating System Installation	24
Disable Incoming Network Access	24
Install Operating System from Vendor Media	25
Harden the Operating System	26
Secure User Accounts	26
Tighten File Permissions	27
Disable Network Services by Default	28
Configure Better Logging	29
Update Operating System	30
Enable Incoming Network Access	30
Secure Applications	30
Application File Location	31
Access to Other Servers	31
Users and Trust	31
Passwords	32
Distributing Passwords	32
Managing Passwords	32
Vulnerability Analysis Tools	33
Intrusion Detection Systems	34
Kernel and Behavior Monitoring	34
File Integrity Checking	35
Network Security	35
Firewalls	35
NIDS	35
Insecure Communication Channels	36

Incident Response	36
Incident Identification	37
Investigation and Analysis	37
Containment and Remediation	38
Restoration	38
Documentation and Review	38
Disaster Recovery	38
Areas to Protect	39
Determine What Happened	39
Phases	40
Identify Disaster	40
Assemble Team	40
Recover Functions	40
Restore Full Service	40
Policy Compliance Monitoring	41
Computer Systems	41
Computer Usage	42
User Awareness and Training	42
Conclusion	42

Chapter 2:

Infrastructure and Data Security

Information systems security is important to businesses and governments around the world. Small businesses, corporations, and governments store crucial data on systems embedded deep within networks, as well as on the Internet's front lines such as on Web servers and email systems. UNIX systems especially have a long history of serving essential roles. Linux and UNIX systems provide infrastructure-level services (e.g., DNS, routing) and play an important role in e-commerce.

Several core security principles exist for securing servers, including least privilege, deny-by-default, and security-in-depth. In UNIX security, you can use a series of tested and proven practices, which I discuss in this chapter, to directly apply each of the principles. When relevant, I include information about how security affects business continuity and disaster recovery. Although I discuss disaster recovery separately at the end of the chapter, you need to integrate disaster recovery into all your security solutions.

The Security Policy

The first step in properly securing a UNIX installation is to develop a security policy that best addresses your organization's needs. Many companies hire consultants or rely on in-house expertise to develop a security policy. Companies that want to follow a standardized set of policy definitions can use free or commercial policy creation software packages to help develop a policy.

Organizations such as SANS (<http://www.sans.org>) and USENIX (<http://www.usenix.org>) offer free security policies. Most commercial software packages walk a security administrator through a series of questions, then create a policy's first draft based on the answers. The software creates a template that the security administrator can use to better tune the policy to the organization's needs.

In this chapter, I focus on a high-level security policy that encompasses a UNIX environment's core needs. Although I don't delve deeply into such a policy's details, the document I use as an example provides a good template for a real-world security policy.

Policy Summary

The policy summary includes the security policy's statement of purpose and the scope to which the policy applies. In my example, which Figure 1 shows, the security policy's purpose is to enforce a predefined set of behavior for systems managers when administering a UNIX server. This policy summary could also include end user requirements and requirements for network connectivity and security. However, your policies should be specific for your infrastructure's key elements.

The policy's scope typically specifies the systems, devices, or information that the policy covers. In my example, the policy's scope includes all the UNIX servers and workstations that the IT department manages.

Figure 1:
Example Policy Summary

Policy Summary

Purpose
This policy defines a set of guidelines for systems managers to follow when installing and managing UNIX systems. The policy's goal is to provide a consistent, verifiable set of configurations across the enterprise for all UNIX systems so that managers can monitor and audit servers to ensure the highest level of security.
To ensure that systems are verifiable as secure and to meet auditing requirements, monitoring must occur for at least user authentication and computer use, unusual log file entries, and file integrity. When possible, use automated monitoring solutions in conjunction with Network Intrusion Detection Systems (NIDSs) and Host IDSs (HIDSs).

Scope
This policy applies to all the company UNIX servers and workstations that the IT department manages.

Responsible Parties

The next section in the policy defines who is responsible for policy enforcement. Charging a team or group with a responsibility without giving them the required level of authority is a certain path to failure. Thus, the policy's Responsible Parties section must define a party that has both the responsibility and the authority to take action. Upper management must give the responsible party the right to directly punish or refer for punishment anyone who violates the policy.



Note

I once worked for a company that asked its systems administrators to write an information security policy. After we wrote the policy and worked with management to make revisions, we came to an impasse because management wanted administrators to be responsible for enforcing the policy but wouldn't give the administrators the authority to chastise or put on report those who failed to comply with the policy. Thus, administrators had no power over policy violators.

In my example, which Figure 2 shows, the responsible party is the Operating Systems group. This group is working to implement and monitor the security that the policy requires. In some cases, enforcement responsibility might fall to a dedicated information systems security or auditors team.

Figure 2:
Example Responsible Parties

Responsible Parties

The Operating Systems group will be responsible for following and enforcing the defined policy. Any policy violations will be reported to the Operating Systems group management team for review. The Operating Systems group has the power to report policy violators to Information Systems Security (ISS) management.

To ensure compliance, ISS management will perform annual audits of the systems that the Operating Systems group manages. Violations will result in the violator being placed on report or being terminated, as necessary.

If conflicts or disagreements arise between the Operating Systems group and ISS management, ISS decisions take priority.

Policy

The security policy's most important element is the actual set of policies or guidelines to be followed. In many organizations, this section contains highly detailed do's and don'ts. In my example, which Figure 3 shows, I keep the policy section only detailed enough to form an initial template so that we can discuss major problems yet remain concise.

Figure 3:
Example Policy

Policy

System Installation

All server and workstation installations of UNIX will follow the standard practice of being removed from the network and having the OS installed and hardened, unnecessary software removed or disabled, and sufficient logging enabled.

Systems Management

Administrators will provide adequate attention to ensuring that file permissions are secure, unnecessary accounts aren't created, a strong password policy is enforced, and proper auditing is performed.

All UNIX systems will be integrated into the company's comprehensive password management software. The software will be used to enforce password strength requirements, as well as to delegate authority for password resets from the Operating Systems group to the Help desk.

All UNIX systems connected to the network will have their internal UNIX-based firewall enabled. By default, all access will be denied except when necessary to allow access from users to the UNIX application being provided. In addition, `tcp_wrappers` will be configured to further protect network services on the UNIX system.

UNIX workstations will have all network services disabled by default; a systems administrator will manually enable only the necessary services. Users aren't allowed to enable network services on their workstations.

All UNIX servers will be installed with file integrity checking tools. These tools will run regularly to determine whether unauthorized changes have been made to system or application files.

This policy provides a set of guidelines rather than specific requirements. In many situations, especially in large organizations, you might want to be specific (e.g., define the software to use to monitor logs and perform file integrity checking).

For a large collection of sample policies, see the SANS Security Policy Project (<http://www.sans.org/resources/policies>). For example, SANS has a password policy (http://www.sans.org/resources/policies/password_policy.pdf) that you could use to further strengthen my sample policy's password requirements.

Physical Security

Businesses often focus on the security of their UNIX OSs, applications, and data but neglect physical security. Physical security encompasses tangible items such as server and network hardware, server closets and rooms, cages and racks at collocation facilities, network cabling in conduits, and (most often ignored) storage areas for backup tapes. Ignoring physical security is dangerous because attackers are often insiders. In addition, even an outside attacker can often more easily walk into a company and gain unauthorized access to data or systems than attack the same company over the Internet or through dial-up access.

Physical security also plays an important role in disaster recovery. Disaster recovery involves quickly bringing back online key services to ensure business continuity. Physical security and disaster recovery go hand in hand because many physical controls can affect systems' survival during natural or man-made disasters. When planning your company's physical security, keep in mind events such as burglary, fire, tornado, hurricane, and loss of facility access.

To construct an effective physical security plan, define the systems you need to protect, the security perimeter around your systems, the threats you need to consider, and the possible defenses against those threats. As you consider your physical security needs, rate your systems by priority and direct the majority of your budget toward protecting your most important systems. For example, you need to keep network services, such as UNIX-based DNS servers, routers, firewalls, and database servers, under lock and key and in a location with fire suppressants. You can usually loosen your security requirements for less crucial systems, such as workstations (except in certain situations, such as on a military base). Although workstations are important to your network, you can easily replace workstations in case of theft or damage. Larger, more expensive server and network hardware are more difficult to replace.

For optimal physical security, you need to consult a physical security expert rather than rely on in-house expertise. Although systems managers and systems administrators are typically adept at securing UNIX systems, physical security requires a different set of requirements.

System Security

The most important aspect of security to UNIX systems managers is system security. Implementing system security in a UNIX environment is difficult for several reasons. First, many environments employ several UNIX flavors, ranging from Solaris and AIX to HP-UX. Each of these systems implements security differently. Fortunately, all UNIX systems share certain qualities (e.g., being file-centric). Thus, many differences in UNIX systems vanish when you view the larger picture of UNIX security, leaving the similarities in focus. However, you still need to focus on the differences as well as the similarities. For example, password management can require drastically different solutions as you cross between UNIX versions.

As I mentioned in Chapter 1, this ebook focuses on the unified management approach to managing UNIX systems. Unified management involves providing consistency across disparate UNIX systems rather than requiring just one UNIX flavor. In terms of security, unified management requires you to build a consistent set of procedures for tasks such as securing servers during OS and application installation and building or purchasing a comprehensive solution for user and password management, monitoring logs and computer usage, and performing other management and security functions.

Operating System Installation

One of a systems administrator's core responsibilities is to install and configure UNIX systems. The systems manager must define and enforce a well-documented set of guidelines for systems administrators to follow when performing installations, because system security begins with a secure installation. In this section I focus on the most secure methods for installing UNIX. These methods vary slightly based on the version of UNIX you're supporting. However, the principles remain the same. To install UNIX, a systems administrator must disable incoming network access, install the OS from the vendor's media, harden the OS, disable network services, configure better logging, update the OS, and enable incoming network access.

Disable Incoming Network Access

Remotely attacking a system that isn't part of a network is quite difficult. Thus, you need to completely disconnect a system from the network before installing UNIX on the system. Doing so creates an *air gap*—the only thing connecting the UNIX system to the network is the air between them (i.e., no connection exists).

If you don't disconnect a new system when you install the OS, an attacker's machine or an already compromised system might detect the newly installed server (e.g., by using a ping sweep to locate new systems, then performing a port scan to detect network services that are enabled by default when the system is installed) and attempt to exploit the unpatched OS. If a system is compromised this early in an installation and you haven't installed file integrity tools, the only way you can determine that the system is no longer secure is to verify the installed files against the files on the vendor's installation media.



Note

Many people forget that modems are commonly found on servers. Some vendors ship servers with a *getty* process listening to the line, which creates a potential avenue for attack. When you unplug network cables, be sure to unplug all the cables—including telephone lines.

UNIX systems are often installed in a secured, mini-LAN that is dedicated to new installations. However, even these environments are vulnerable to attack. In fact, these machines often fall victim to attack faster than servers on the production network because many of the machines in these so-called secured LANs are left half-configured. If you decide to use a mini-LAN dedicated to new installations, be sure to completely restrict access to the Internet and the production network to outgoing access only—and even then only to servers or Internet sites necessary for updating newly installed UNIX systems. This action adds considerable security to the mini-LAN installation environment.

Install Operating System from Vendor Media

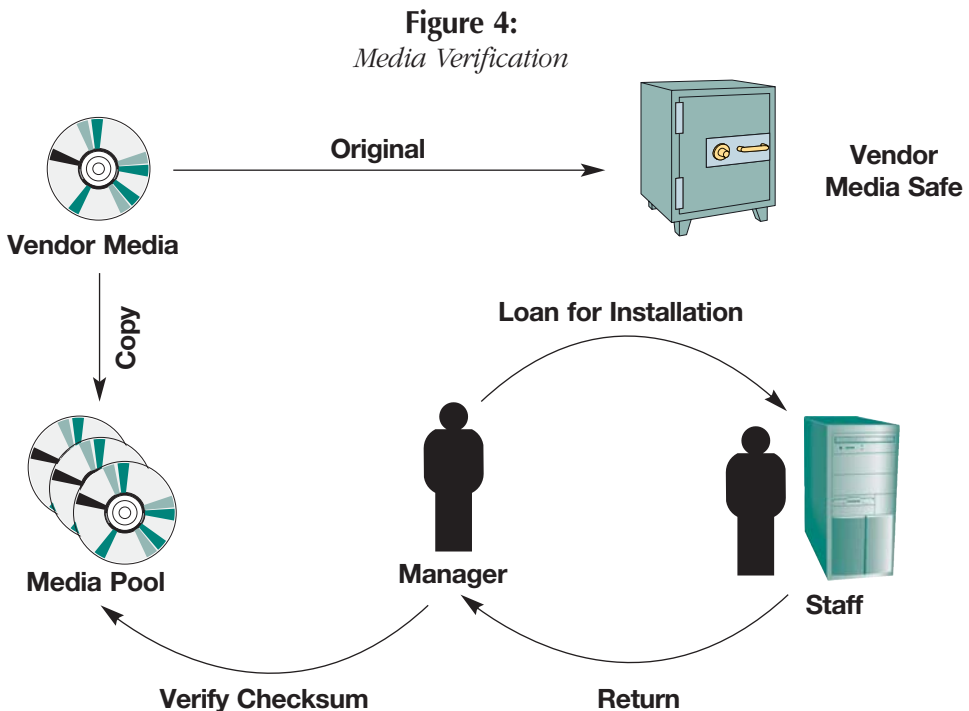
The next step is to actually install the OS from the vendor media. In most cases you'll use a copy of the media, which isn't necessarily a best practice but makes sense so that the original media isn't lost or stolen. You typically make a copy of the vendor media, then store the original in a safe or offsite location in case of a disaster.

The caveat is that using a copy of the vendor media exposes you to the risk that the copy has been tampered with. Although this risk is small in most organizations, it is still real. As you develop your procedures, keep in mind that most attacks are by insiders. One of the easiest ways for an insider to compromise a system is to do so before any security tools are installed and configured (i.e., when the OS is installed). Using verified media to perform installations minimizes this risk.



Note

Determining how to secure your media is an important concern. A good plan is to assign a manager to monitor the installation media and maintain a set of checksums. As Figure 4 shows, when the manager releases the media, he or she documents who receives the CDs or tapes. When the staff member returns the media, the manager can quickly verify that the checksum hasn't changed and thus verify that the media is still valid. Although this level of control might seem tedious, ensuring that your media is secure is important. Otherwise, later monitoring and integrity checking might be too late.



Another consideration when installing a new UNIX system is that vendors have a bad habit of installing too much software and enabling too many services by default. Increasing the amount of software and number of services a server provides greatly increases the server's vulnerability footprint. For example, most Linux distributions are well known for installing the Apache Web server by default even though attackers have significantly exploited Apache vulnerabilities in the past. Thus, you shouldn't install Apache unless necessary. Systems managers need to refrain from increasing servers' risk without justifiable cause. Vendors are slowly adopting a more minimalist stance than in the past and are including less software and disabling more services during a default installation. OpenBSD, an open-source BSD UNIX, takes this stance to an extreme and disables all services except Secure Shell (SSH).

Harden the Operating System

Hardening a server means performing a series of changes to a system to better secure the system. Hardening usually takes place during OS installation but also occurs as you develop new procedures, install new patches, and deploy new applications.



Note

You should always perform a new installation when you receive a new server or when a server with a preexisting OS comes under your management.

When hardening a server, you need to keep in mind the three principles I mentioned earlier: security-in-depth, least privilege, and deny-by-default. Security-in-depth means putting as many obstacles as possible between an attacker and potential targets. Least privilege means giving users only the level of access (i.e., privileges) they need to perform their jobs. Deny-by-default means that when you configure services, firewalls, or other security barriers, you should deny all incoming and outgoing services unless explicitly allowed.



Note

The alternative to deny-by-default is to allow all incoming and outgoing services by default unless explicitly denied. Although allow-by-default makes managing services easier than deny-by-default, allow-by-default makes maintaining security more difficult because you must constantly update a list of dangerous services, ports, and IP addresses to deny. Using deny-by-default is a best practice because a deny-by-default configuration's fail-safe mode is to deny service, thus protecting your UNIX systems in the event of a failure or a bad security barrier configuration (e.g., a firewall). Unfortunately, most UNIX systems installed with vendor defaults don't have the deny-by-default configuration. I discuss this problem throughout the chapter.

Secure User Accounts

Vendors often include unnecessary accounts on servers. These seldom-used accounts include FTP, uucp, guest, and news. You might encounter others as well, depending on your UNIX vendor. These unnecessary accounts exist because vendors often use default settings to build systems and therefore install unnecessary software. The easiest solution is to disable or delete the accounts.

Another concern when securing a system is knowing who has administrative account access. Special accounts, such as `sys` and `root`, have access to crucial OS files. The most powerful and therefore the most dangerous special account is `root`. Systems managers must define policies to control access to and monitor usage of the `root` account (or any account with a User ID—UID of 0).

You need to use multiple layers of security to control access to the `root` account. The first layer, and the most commonly used strategy, is to not give the `root` password to staff that don't need access to the account. In addition, you shouldn't allow `root` access directly from the network. Because most UNIX environments use SSH, compromising the `root` password over the network isn't the issue—but auditing `root` usage is difficult if administrators log on directly as `root`. You need to disable `root` logons in remote-access software (e.g., SSH) and require that administrators use the `su` command (or `sudo`, which I discuss) to become `root`.

If possible, you need to entirely disallow direct access to the `root` account. Use role-based access control (RBAC) if your OS supports it. Solaris supports a limited form of RBAC; you can delegate certain OS rights to specific users. Those users then have `root`-like privileges for the functions you specify. RBAC doesn't provide a comprehensive solution for delegating rights, because no general-purpose UNIX systems fully implement RBAC. Add-on solutions such as Symark's PowerBroker (<http://www.symark.com/powerbroker.htm>) offer RBAC features for Linux and UNIX systems. Sudo is a widely used UNIX open-source rights-delegation tool. Sudo lets managers delegate rights in a more fine-grained fashion than giving everyone access to the `root` account. Sudo uses `syslog` to log every command that runs, as well as who ran the command. Sudo is an excellent tool for auditors; you should mandate its use for administrative functions in your environment.



Note

You need to keep the logs that RBAC and `sudo` generate longer than you keep regular server logs. You might not discover inappropriate privilege use for several weeks or months, so you need to be able to retrieve usage records for long time periods. If you don't keep these logs, you might have a difficult time finding credible evidence to prove privilege abuse.

Tighten File Permissions

As I mentioned in Chapter 1, UNIX is a file-centric OS. You use files to access everything from application configuration information to hardware devices. Because file permissions play an integral role in UNIX security, ensuring proper file permissions maintenance must be one of your security procedures' major focuses. You need to train your systems administrators to consider file permissions one of the most important elements of system security.

File and directory permissions modes can inform the OS of special modes to use when executing a file. An executable file can have two special modes: set group ID (SGID) and set user ID (SUID). With SGID the executable runs with the permissions of the group owner as opposed to the user running the program. With SUID the executable runs with the permissions of the file owner. Most host security analysis programs, which I discuss later, automatically search for and report SUID programs.

A common use of permissions in UNIX is to grant or deny read (r), write (w), or execute (x) access to a file or directory. UNIX has three groups of security permissions: user (u), group (g), and other (o). The user is the UNIX account that owns the file, group is the UNIX group that has group

ownership of the file, and other is any user who isn't an owner or in the group that owns the file. In general, you should use the deny-by-default and least privilege principles to create files, which gives only the owner access. If necessary, you can also give the owning group access to key files. This strategy is useful when users share files. The other group should have access to only certain files—and in most cases (e.g., the `/etc/passwd` file) the access should be read only.

The `umask` system tool defines default security permissions for newly created files and directories. Settings are defined in a script that runs when the user logs on (e.g., in `/etc/cshrc`). Many remote file access tools, such as SSH File Transfer Protocol (SFTP), also need proper configuration to apply secure permissions when creating or copying files and directories.



Note

Ensuring that system and application files have the proper security permissions is on par with maintaining a strong password policy. Strong security permissions can protect you if a server account is breached, because most accounts wouldn't be able to make unauthorized changes to key files.

The way that UNIX traditionally implements file security is sufficient in most cases, but this method can present a challenge for administrators who want to implement fine-grained access control. Many UNIX systems, including Linux, now include ACLs as a complement to traditional file security. ACLs let administrators and users specify a list of users and their access rights, rather than restricting access control to the more limited user-group-other model. ACLs greatly simplify UNIX security administration.

Disable Network Services by Default

The best practice in terms of network services is to disable all services during system installation, then activate only the services you need. Decreasing the number of services a server offers lessens the server's vulnerability footprint, thus limiting possible avenues for attack. This principle is especially true for the services typically enabled on a default installation (e.g., FTP, Telnet), because the server software that offers these services has a history of exploits.

UNIX services usually start in one of two ways: from a startup script or through the Internet superdaemon `inetd`. To disable a service in `inetd`, remove or comment out the service line in `/etc/inetd.conf`. Following are examples of a service that is enabled and disabled.

```
#ftp    stream  tcp     nowait  root    /usr/libexec/ftpd    ftpd -l
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd    ftpd -l
```

Services commonly started from `/etc/inetd.conf` include Telnet, FTP, and Samba.

`Inetd` has a long UNIX history but is slowly being replaced by `xinetd`. `Xinetd` uses each service's configuration file to define the service rather than configuring all services in one central configuration file. For example, for `xinetd` the `ftpd` service would probably be defined in `/etc/xinetd.d/ftpd`, whereas in `inetd` the `ftpd` service would be defined in `/etc/inetd.conf`. To disable a `xinetd` service, change the configuration line *disable = no* to *disable = yes* in the appropriate `/etc/xinetd.d` configuration file. `Xinetd` also lets you specify which hosts can access the service, much like

tcp_wrappers works. (I discuss tcp_wrappers later.) Xinetd gives you an additional tool in your security-in-depth arsenal.

The location of startup scripts depends on whether the UNIX you're using is based on BSD or System V (SysV). For BSD systems startup of most services begins with /etc/rc, whereas for SysV systems startup begins via a script in /etc/rc.d/init.d. In BSD you can comment out the relevant lines in /etc/rc to disable services, whereas in SysV you can use chmod to disable execute permission, as in the following example.

```
# chmod ugo-x /etc/rc.d/init.d/samba.sh
```

You need to restrict access to services that you want to leave enabled. Three methods let you restrict access to network services: firewall, application, and tcp_wrappers. With a firewall you can disable access at the TCP/IP protocol's packet layer. Tcp_wrappers, which I discuss below, provides a UNIX method of allowing or disallowing remote client access to local services. Finally, many network services let you specify which hosts have access. Samba and Apache are example of these services; both let administrators specify which remote hosts can connect to the service.

Wietse Venema's tcp_wrappers protects TCP-based network services by letting administrators specify which remote hosts are granted or denied access to the network port the network service is using. Tcp_wrappers' most useful feature is that the service typically runs independently of the service being wrapped. Thus, the wrapped application doesn't need to know that tcp_wrappers is running it. Wrapping a service in /etc/inetd.conf's configuration line lets you configure the service for tcp_wrappers protection. In addition, you can compile and link certain applications, such as the Apache Web server, with tcp_wrappers support. Although Web servers don't generally run from inetd, this feature lets them rely on one tcp_wrappers configuration for network access control.

Although I don't cover tcp_wrappers in depth, you need to know that its prevalence on UNIX systems makes it an excellent method for consistently protecting network services. With the proper configuration, you can enable logging of all network service access, such as SSH. This feature is especially beneficial when a server's logs are linked with a monitoring system or IDS.

Configure Better Logging

An attacker's dream is lax logging. An auditor's dream is strict logging. Of the two, satisfying the auditor makes the most sense. System logging in UNIX is primarily syslog's realm. Syslog is typically implemented as a daemon that starts when the system boots up (i.e., syslogd).

If you're running multiple UNIX servers, you need to log to local log files and a central log host. To configure this logging in /etc/syslogd.conf, specifying that all log entries send to the remote host, as in the following example.

```
* @logger.example.com
```

In this example, messages that send to the local syslog daemon also send to logger.example.com. You might want to restrict sent messages to only those related to security. However, a good practice is to log everything, then post-filter the results to drill down to the data you need. This method gives systems administrators access to all log entries on one central server, and systems manager and auditors have a complete history of the actions performed on managed servers. As I discussed earlier, any commands that sudo invokes log to syslog. Thus, using a central log server lets you build one monitoring application to monitor privileged access.

A central log server offers many additional capabilities. For example, a systems administrator or automated monitoring application can scan the collected logs to determine whether a pattern of access on several systems might indicate an attack. If you inspect logs only on a server-by-server basis, you might consider an anomaly an isolated incident rather than notice a pattern.

In observance of the deny-by-default principle, on most UNIX servers you need to disable the syslog daemon's ability to accept log entries over the network. For Linux's `syslogd`, you can add the `-s` option. To completely stop the local `syslogd` daemon from sending messages over the network, you can add `-ss`.



Note

A common error when first configuring a central logging host is to accidentally enable remote logging on the logging host itself. Doing so causes the log host to send itself a log entry, which the log host sends itself again, and so on. Most syslog daemons aren't intelligent enough to stop this feedback, and the logs quickly fill up. Pay close attention when configuring these settings.

Just logging system events isn't enough. You must also monitor the logs and ensure that recorded events don't indicate noncompliance with a policy. At the end of this chapter, I discuss how to monitor for policy compliance and detect attacks and compromises.

Update Operating System

Vendors typically release new installation media only after several major updates to their OSs are released, so the OS installed from the media might be outdated. Therefore, the next step is to update the OS to the most recent patch level. (I discuss patch management in more detail in Chapter 4.)

Placing an unpatched server on the network is dangerous. Even if you disable all the network services, an unpatched kernel can give potential attackers a Denial of Service (DoS) opportunity. After you patch a server, you need to review the changes and possibly reapply the fixes you applied during the initial hardening. You need to repeat this cycle after each update: Update the server, then harden it. Systems managers benefit from automating all or most of the hardening process by using vendor-supplied tools or open-source or commercial solutions, or by developing in-house scripts to harden their specific versions of UNIX and the applications they provide.

Enable Incoming Network Access

After you've secured the server, you're ready to provide service to the network. You've installed the OS, hardened the server, and applied the most recent security patches. The final step is then to enable incoming network access.

Secure Applications

After you install a server, your focus changes from the UNIX OS to the target application. Applications are even more vulnerable to attack than are the underlying OSs. This security vulnerability exists because of improperly written UNIX software and inappropriate privilege use that violates the least privilege principle.



Note

Applications often run as root when they could easily run as a normal user. This scenario occurs because the application needs to listen on a privileged port (i.e., any network port between 1 and 1024), and the application must therefore start as root. However, after the application gains control of the port the application can drop its privileged status as root and change to another user.

You need to be careful when installing and configuring applications on your UNIX servers. Your installation procedure documentation must consider the following:

- Application program files', configuration files', and data files' locations
- Whether the application requires access to another server for operation
- The kinds of users the application will support and your level of trust in those users

I review each of these areas in more detail in the following sections. Your installation procedures need to address each area and provide guidelines for each supported application.

Application File Location

In general, UNIX applications follow one of two installation methods: Application files are in a directory specific to the application, or application files are spread across directories in /usr/local. A recent trend has been to install applications in /opt, but this approach is just a variation of the first method I mentioned. From a management standpoint, installing an application in its own directory is the better of the two options. This method lets you easily monitor for file changes (e.g., using file integrity monitoring software) and enable simple backups and restores. In addition, depending on how fast the application data files will grow, dedicating an entire file system to the application is a good idea—and having all the files in one location makes creating such a file system easy.

Access to Other Servers

Another consideration is whether the application requires access to other servers for operation. Many applications (e.g., Apache) require at least DNS access. In addition, many enterprise Apache applications are built using a scripting language embedded in the Web pages and require access to database services from Oracle, Sybase, or the open-source MySQL relational database management system (RDBMS). Therefore, as part of the application installation and configuration procedure, you must open access to these services to the Apache server. As always, you need to follow the least privilege principle and give Apache only the minimum level of access necessary for the Web application to function.

Users and Trust

Finally, you need to study the target user base for the application you're deploying, with a special emphasis on determining the danger of attack. For a Web application that is based on Apache and is available to the Internet community, the danger of attack is high. For a small, targeted application dedicated to a receiving department, the danger might be low. Although systems managers want to

provide maximum security for all their applications, organizations have finite budgets allocated to security. Thus, you need to allocate the appropriate amount of resources for security depending on how crucial the target application is to your organization's operation and the danger that those with access to the application present.

Passwords

Users enter passwords to log on to servers and access applications. Most UNIX systems store account information in `/etc/passwd` and encrypted passwords and additional information in a shadow password file. When a user logs on to the server, the system scans the password database and verifies that the entered password matches the password in the database. This system is sufficient in many environments because it's simple to maintain if you're managing only one or two servers and because you can easily back up and restore the password and shadow password files. More complex environments require a more comprehensive system to distribute and manage passwords.

Distributing Passwords

Distributing account information, including passwords, has long been Network Information Service's (NIS's) realm. NIS is a set of protocols and software that Sun Microsystems developed to distribute UNIX configuration information in large networks. UNIX has slowly outgrown NIS, but NIS still exists in many environments. Unfortunately, NIS is insecure; you should use it only in a highly trusted network.

Lightweight Directory Access Protocol (LDAP) is a method of centralizing passwords that is finding its way into UNIX networks. LDAP over Secure Sockets Layer/Transport Layer Security (SSL/TLS) provides a secure protocol for verifying passwords. You can expand LDAP over SSL/TLS to include configuration information for everything from printers to applications. Use LDAP whenever possible.

How to support LDAP (or any authentication framework) varies across UNIX platforms. For systems that use the Pluggable Authentication Modules (PAM) interface (e.g., Solaris, Linux, FreeBSD), support can be as simple as configuring your servers to use LDAP PAM. For other UNIX systems, you might need to purchase commercial software or locate open-source solutions from your vendor or other users.



Note

Sun developed a new version of NIS known as NIS+. NIS+ addresses many of NIS's problems. However, NIS+ isn't as widely implemented and deployed as NIS. To use NIS+, you must restrict your supported UNIX systems to Solaris. (Linux technically supports NIS+, but the support is buggy and not actively developed.) Move to LDAP if possible because of its wide industry support.

Managing Passwords

Enforcing a solid password policy is important. Two main attacks against weak passwords are remote access attacks and password file attacks. In recent years, defense against these attacks has improved,

in the form of temporary account lockouts for remote access attacks and shadow passwords for password file attacks.



Note

Surprisingly, for many years Linux systems offered shadow passwords and MD5 hashing as an option rather than the default. Most Linux systems now offer shadow passwords and MD5 hashing as the default. If your Linux or UNIX systems don't use shadow passwords and MD5 as the default, you need to include in your policy a requirement to enable their use.

Another aspect of password management is ensuring that users pick strong passwords. Enforcing this behavior in a UNIX environment can be difficult because many UNIX environments support multiple flavors of UNIX. Having multiple flavors lets systems managers best support their organization's mix of needs and requirements but makes defining and enforcing a set of password policies difficult because managers must customize their password procedures for every target UNIX flavor.

The best solution in a mixed environment is to use a password management tool that plugs into each target system. Although some organizations use customized scripts and Web applications as password management solutions, often as your organization grows you'll want to purchase and deploy one of the large identity and password management applications designed for the task (e.g., NetIQ's VigilEnt Password Manager—<http://www.netiq.com>, Computer Associates' eTrust Admin—<http://www.ca.com>). A benefit of comprehensive password management solutions is that they typically offer self-service to users who need to reset their passwords. These solutions also give the Help desk the ability to delegate password management.

Vulnerability Analysis Tools

Vulnerability analysis tools give you a way to assess vulnerabilities on your network and servers. These tools, combined with a rapid response team that can reduce or eliminate vulnerabilities, enhance your vulnerability management ability.

Network vulnerability scanning requires that the scanner first detect which services are available on a server. Next, the scanner maps the ports found with the banners retrieved (a banner is the greeting a network server provides when you connect to its port) to match against a database of known applications. The scanner then uses this information to perform a series of tests against the services that are usually specific to the application the scanner determined is running. Common tests include buffer overruns and DoS attacks and can also include preprogrammed requests and responses known to result in compromises in vulnerable versions of the software.



Note

In general, randomly performing network scanning and vulnerability testing on a production server isn't advisable because the scanning might cause the server to go offline accidentally (e.g., during a test for DoS).

Examples of network vulnerability analysis tools include the open-source Nessus (the most popular network vulnerability analysis tool currently in use) and commercial scanners from

companies such as Internet Security Systems (ISS—<http://www.iss.net>). Both of these tools help you detect services and find vulnerabilities across all of your UNIX servers. Depending on your needs, Nessus and other open-source tools usually provide adequate functionality. The primary differentiator between Nessus and commercial products isn't the scanning quality but rather the reporting capability. Nessus works well for single hosts or targeted network scanning, but commercial products tend to provide more readable reports for large networks.

In addition to network vulnerability analysis tools, you can use software that you install on a server and use to detect vulnerabilities in local software. This type of software is more intrusive than network vulnerability analysis tools but often provides more useful information and returns fewer false positives. Examples of open-source host vulnerability analysis tools include Tiger for UNIX and Bastille for Linux (Bastille detects and corrects many security problems). You need to understand that most host vulnerability analysis tools focus on obvious security risks such as insecure file permissions, enabled services that are known to be insecure, and unnecessary default accounts (e.g., the FTP user account). Because these areas are the most prone to result in a compromise, the vulnerability scan and subsequent hardening can dramatically increase your servers' security.

Vulnerability scanners that are specific to Web applications also exist. These tools comb a Web application, looking for potential and known security problems. Regardless of how safe a Web application programming language purports to be, you need to run these tools against Web sites before deployment and regularly thereafter to find new holes. As with host scanning software, open-source and commercial solutions exist. Most of these scanners target Common Gateway Interface (CGI) scripts, known Apache and other Web servers' default installation files, and known problems with languages such as PHP, Active Server Pages (ASP—running under ChilliSoft ASP), and ColdFusion.

Intrusion Detection Systems

IDSs are relatively common in today's networks. IDSs fall into two categories: HIDSs and NIDSs. NIDSs receive the most media attention, although HIDSs also give systems managers a lot of power and manageability. The term HIDS actually encompasses several technologies, some of which existed before the more comprehensive concepts of NIDS and HIDS became popular. Two common HIDS components are kernel and behavior monitoring and file integrity checking.

Kernel and Behavior Monitoring

Kernel and behavior monitoring usually requires the monitored server to have a special kernel or kernel module loaded. The modified kernel monitors for suspicious activity, such as consistent attempts at unauthorized access to special privileges or files, new modules being loaded, or applications that behave differently than their normal baseline.

Although kernel and behavior monitoring can play a key role in your IDS strategy, getting this type of monitoring to work often involves a large initial time commitment. In general, you need a baseline analysis of typical behavior. Kernel and behavior monitoring IDS is more proactive than file integrity checking, which is an after-the-fact check. Kernel and behavior monitoring can help you stop an activity before it proceeds further.

File Integrity Checking

File integrity checking plays a pivotal role in a secured environment and is part of a complete IDS. You need to provide file integrity checking on all your UNIX servers. Even if you determine that a full IDS isn't required, you should still require file integrity checking on every server. When used with vigilant log monitoring, file integrity checking gives systems and security administrators timely information about attempted, in-progress, and successful system compromises.

A file integrity checker compares a file on a file system to a database containing information about the file's Last Known Good state. The checker determines whether a file has changed. If the file on disk and the file information in the database are different, an alert generates. File integrity tools typically use a combination of hashing functions, such as MD5, to compare file contents, and metadata, such as ownership and inode number, when comparing files. One of the most common file integrity tools is Tripwire, which began life in the UNIX world but has also expanded into Windows. Several solid open-source file integrity tools exist (e.g., Advanced Intrusion Detection Environment—AIDE).

A problem with file integrity checking is that an attacker might alter the file database before the tool runs. In this case the tool won't detect or report any changes because the database was updated. To guard against this possibility, you need to store the file database on a secured remote server or on nonwritable media such as a burned CD-ROM.

Network Security

Firewalls

Firewalls provide a security barrier between an internal network or hosts and external network devices. Two types of firewalls exist: packet filtering and application level. The traditional firewall device is packet filtering, which scans incoming IP packets and determines whether to allow a packet in or out. Packet filtering firewalls have the benefit of speed but don't understand the application protocol and therefore can't filter packets based on information such as incorrect application protocol usage (e.g., a bad HTTP header). Application level firewalls work at a higher stack layer than packet filtering firewalls and therefore understand the application protocols that are in use. However, packet filtering firewalls are slower and are more processor intensive than application level firewalls.

You use a firewall when you need a barrier between different areas of trust in your UNIX environment. An obvious example is the interface between your corporate network and the Internet. Additional areas include between departments and IT test labs.

Many UNIX OSs include firewall software in the kernel. For example, you can use Solaris and Linux as firewalls. According to the security-in-depth principle, you also need to use the packet filtering firewalls built into these systems as a way to protect servers. For example, a Solaris Oracle server should use a firewall that filters all traffic except incoming requests and outgoing responses. This practice gives the administrator more time to correct configuration errors (e.g., enabling RSH), because the service is inaccessible even if enabled.

NIDS

A NIDS constantly monitors network traffic, looking for signs that a network-based attack is in progress. NIDSs can typically monitor for known attacks (signature based) and monitor for behavior that indicates an attack might be in progress.

Even the best-configured NIDS sometimes generates false positives (i.e., alerts caused by activities mistakenly marked as attacks). This problem leads some systems managers to question whether they should use NIDS solutions for all their installations. One of NIDS's (or any IDS's) major problems is the amount of tuning necessary during the initial installation and the time necessary to monitor output. However, the general consensus is that if your organization has the resources to deploy and monitor a NIDS, you should do so. Otherwise, expend your efforts in other areas, such as increased server hardening and installing file integrity checking tools on your systems.

Insecure Communication Channels

One of the most important principles of network access is that communication channels must be secure. When considering administrative and end user remote access to a server, you need to use a cryptographically secure channel. Also, verify the identity of all parties involved in a channel. Some attacks are easy if just one network point is compromised.

An example of an attack possible because of an unsecured channel involves a compromised DNS server. (And unfortunately, the most common DNS server, BIND, has a history of vulnerabilities.) DNS drives most UNIX environments; when an administrator connects to a remote server, he or she typically uses a domain name, such as `ldap.example.com`. If an attacker has control of the DNS server or poisoned the server's DNS cache, the attacker can redirect administrators to a server that the attacker owns. Even if the administrator were using an encrypted channel, he or she might supply a logon and password to the remote server to gain access—the attacker would then immediately have access to this information. Depending on the attack's timing, the attacker could attack the real server while the administrator was trying to determine why his or her logon failed.

In such a case, technology such as SSH or certifications can verify the remote computer. Because the attacker owns the remote computer, the SSH host key validation stage would fail, thus alerting the administrator to the problem. Because of DNS's and unverifiable network services' inherent insecurity, you need to rely on alternative verification methods to reduce your risk of attack.

Incident Response

Security incidents often occur in large environments simply because such networks offer attackers a big target. Attacks usually occur in small environments because of random selection or because an attacker was searching for a specific target (e.g., documents at a small research medical clinic). Because most sites eventually suffer a compromise, you need to have an incident response plan in place.

The first step is to define and test policies and procedures to address your response team's needs, which involves writing a security and backup policy and procedure document and assigning incident response team roles. A backup policy and procedure document is vital for incident response and disaster recovery. A major breakdown during later phases of incident response and disaster recovery is failing to quickly restore crucial systems and data to servers.

Assigning team roles is also important. If you don't define team roles now, when an incident occurs your impromptu team will probably fail to accomplish at least one core proper incident response requirement. If you don't expose a team to the predefined procedures, the team will lack coordination when an attack occurs.

In principle, incident response in UNIX environments is similar to other environments. Generally accepted responses after an attack include the following.

1. Incident identification
2. Investigation and analysis
3. Containment and remediation
4. Restoration
5. Documentation and review

Incident Identification

After an incident occurs, you need to react quickly. In some cases, such as when customer information is compromised, you must immediately notify the appropriate authorities. Then, determine whether the incident is in progress or has already occurred. Typically, an IDS notifies the information security group of an incident, a file integrity tool notes a discrepancy, or a manual log file review (e.g., of `syslog's /var/log/messages`) shows an odd log message that the systems or security administrator reviews and determines to be from an attack. Next, you need to notify the incident response team so that the team can follow the proper incident response procedures.

The team typically makes an initial review of the system logs and state to verify that an incident occurred. You need to determine whether an incident involved an attack or resulted from a software or hardware glitch.

First, you might want to run commands such as `last`, `ps`, and `lsof` and save the output to another server. Second, you need to preserve evidence, which is problematic in a production environment because the compromised systems are often in use and taking the systems offline can be expensive. Several options exist for handling this problem. If your UNIX system is running with mirrored storage (e.g., RAID 1), you can break the array and safely store a mirror for later review. Another option is to use a UNIX tool such as `dd` to make a bit-by-bit copy of the media, in which case you need to unmount the file system or at least use the `mount` command to remount the file system as read-only. Finally, you can remove the storage media, replace it with new media, and rebuild the system.



Note

Chain of custody is important if you want to criminally prosecute an attacker. Document each stage of your incident response, including who handled any drives or storage media containing evidence.

Investigation and Analysis

At this point the incident response team is ready to investigate the incident in more detail. Depending on your business needs, the investigation might be detail oriented or mainly concerned with quickly identifying the exploited vulnerability to fix other systems. Investigations should be thorough enough to locate all affected systems, determine which systems have the vulnerability, and put into place a restoration and patch plan.

The nature of UNIX logging gives UNIX managers a good chance of finding a series of logged events across the network. If you've configured central logging, review the centralized logs for traces of odd activities (e.g., suspicious logons).

Investigation and analysis typically includes forensic analysis, which is a detailed examination of the compromised system. Forensic analysis is as much art as science and requires a high degree of skill. If your organization is large enough, train an onsite staff member in computer forensic analysis. This person needs to know how to use tools such as Brian Carrier's open-source The Sleuth Kit and Dan Farmer and Wietse Venema's The Coroner's Toolkit (TCT), as well as commercial packages.

Containment and Remediation

The next stage begins with containing the problem. You might think that this stage should come earlier; however, properly containing an incident before you determine the root causes and affected systems (which you discover during the investigation and analysis stage) is difficult if not impossible.

The team might take several steps in containment, from locking out an affected user account and ensuring that nobody is actively logged on as that user (in UNIX, locking an account doesn't affect currently logged on users or their permissions to continue working and changing files) to taking affected systems offline to ensure they don't assist an attacker in further compromises. Taking affected systems offline is drastic but might be necessary if you can't quickly stop an attacker or remove the exploited vulnerability.

After containment is remediation, in which you plug the holes that allowed the attack. For example, if the exploited vulnerability were in the local FTP server, you'd need to patch the software if a fix were available, or disable FTP access until you developed, tested, and deployed a fix.

Restoration

In this step, restore the system to a Last Known Good state. Simply fixing the problem on the affected systems isn't advisable because the attacker might have planted a Trojan horse on the affected systems. A Trojan horse would give the attacker a back door to later reenter the system. The restoration stage requires a properly defined and executed backup strategy. Disaster recovery planning is beneficial during incident response because the affected systems are a total loss.

You need to restore the system to the point in which it wasn't vulnerable to the attack that compromised it. Otherwise, you might have a difficult time determining whether the attacker exploited the system earlier and possibly left trapdoors to get back into the server. If you can't restore the system to a nonvulnerable state or immediately correct the vulnerability, you need to perform a fresh installation, restore only the data, and lock down the vulnerable service until you properly secure the system.

Documentation and Review

Finally, you need to document the incident. The documentation needs to include the vulnerability used and an explanation for why the existing security policies and procedures didn't correct the vulnerability before the incident occurred. This information is useful for later improvements.

Disaster Recovery

The idea behind disaster recovery is to plan for worst-case scenarios. Disaster recovery is an important element of both a security policy and an incident response procedure. The security policy defines the methods and procedures that protect against and eliminate potential causes of a disaster. Disaster recovery is how you respond to those disasters.

Several tasks make disaster recovery easier: Perform regular backups of data and system configuration information, practice your disaster recovery procedure at least semiannually, and use redundant systems and remote sites that can help you automatically recover from a disaster. In addition, the most important element in disaster recovery is documentation. Documentation that details system installation, from both vendor media and backup software, must be available to the recovery team at all times. UNIX lets you easily provide documentation on most configuration settings as well, because configurations tend to be in text files that you can print and include in server documentation.

You need to codify these and other tasks into a plan. In the following sections I develop a disaster recovery procedure development cycle outline and explain how that procedure addresses a business' needs.

Areas to Protect

The first step is to define the areas that your disaster recovery plan needs to protect. This phase is often called the Vulnerability Assessment or Definition of Requirements. In my example, I focus on the systems under the Operating Systems group management. Chapter 1 defined five levels of importance for these systems, which you can use to create the following sample prioritization.

- Infrastructure servers
- Data servers
- Application servers
- Interactive servers
- Workstations

Because every site is different, no hard and fast rule exists for designating the areas that your disaster recovery plan protects. Depending on your company's workflow, you might need to reprioritize your equipment's importance specifically for disaster recovery. For example, you might want to ensure that a subset of your workstations have the same priority as your infrastructure, data, and application servers, because your users can't work without workstations.

Determine What Happened

After your core services are functional again, you need to determine what caused the disaster. Depending on your staff levels, you might be able to dedicate a second team to determine the disaster's cause, while the disaster recovery response team performs recovery. Although dedicating as many people as possible to recovery might seem like the best approach, in a crisis situation a better option can be to have a small, well-trained team dedicated to disaster recovery rather than a large team trying to attack each stage of the process.



Note

Problem determination is similar to the investigation stage during incident response.

Phases

A simple disaster recovery plan involves four stages. These stages are similar to the phases of incident response. The most important consideration in each stage is to restore crucial services as quickly as possible and restore less-used services as necessary.

Identify Disaster

First, determine what happened. Did you lose servers because of a fire? Did you lose a building? Did you lose key personnel? What services or systems are no longer functioning, and what services must you immediately restore? Identifying the disaster lets you quickly pinpoint the affected area and ensures that you don't leave your servers and equipment in danger.

Assemble Team

When an incident occurs, you need to assemble the team you've previously put together. At least one team member will probably be unavailable. This probability is why cross training and having backup team members is important.



Note

You need to realize that none of your team members might be able to reach the primary site (e.g., if an entire computing facility is lost). Thus, important functions must have an offsite backup available. Banks and hospitals regularly use offsite backup locations. These locations are usually synchronized with the data and services at the primary site. If the primary site fails, the backup site brings key resources such as billing online. Supporting an offsite location can be expensive. UNIX tools such as rsync and replication features built into most databases can ease the burden.

Recover Functions

The next step is to recover minimal service. At the minimal level of service, the company can accomplish its core functions. Most users will be frustrated with a minimal level of service, but the company can survive for a short time at this level.

As an example, let's consider a financial services company. The minimal level of service includes access to market information and trading and to a telephone system. The disaster recovery team must quickly provide access to the Internet, trading system, and customer records (in an emergency situation, a hard copy of client information might be sufficient) and ensure that a telephone system is up and able to process incoming and outgoing calls. Services that users usually consider important, such as email, wouldn't be available at this level.

Restore Full Service

The final goal is to attain a normal level of service. At this level of service, all company services are available and users can access the data and information they need.

One of the best ways to ensure a quick restoration of full service is to use the disaster recovery features built into modern backup systems. You can integrate software from companies such as Veritas into your backup strategy to allow for a quick restoration of servers. The key difference between disaster recovery software and normal backup software is the time necessary to bring a server back online. With backup software, you typically need to restore an OS from vendor media, whereas with disaster recovery software the backup lets you restore the system from scratch. Software to assist in disaster recovery is a wise investment.



Note

UNIX's `dd` can make a byte-by-byte copy of a disk. Many UNIX sites rely on `dd` the same way that Windows sites rely on tools such as Symantec's Norton Ghost. Although `dd` is a powerful tool, it doesn't always work if the disk being copied to isn't the same size as the disk you're copying. To provide quick and reliable restoration services, use disaster recovery software rather than `dd`.

Policy Compliance Monitoring

Policy compliance monitoring is complicated because it encompasses more than just ensuring that systems are installed and managed according to the security policy. Rather, policy compliance is comprehensive monitoring of everything from proper systems management to end user awareness training and programs. The three policy compliance monitoring areas I discuss are computer systems, computer usage, and user awareness and training.

Computer Systems

Monitoring policy compliance on UNIX systems involves matching various monitoring tools against the appropriate areas of your security policy. In the case of the security policy I described earlier, the guidelines cover system installation, systems management, password management, and network security. You can often categorize policy monitoring by the areas your security policy outlines. Categorizing security into sections lets you more easily automate some or all of the policy compliance monitoring.

Monitoring policy compliance during system installation typically means requiring that new servers run against an automated testing application. Most often this application is a mix of a host and network vulnerability analysis tools, which I discussed earlier, along with a series of scripts that monitor for required configuration settings that aren't within the vulnerability analysis tools' scope. For example, if your installation policy requires that all UNIX servers have a deny-by-default configuration of `tcp_wrappers`, your tool needs to ensure that `/etc/hosts.allow` and `/etc/hosts.deny` exist, their permissions are safe (e.g., file permissions mode `0400`, which is very restrictive), and their contents are appropriate for the server. Monitoring policy compliance for a production server is similar. Again, the most useful tactic is to run a set of automated tests against the server on a regular basis, looking for exceptions that violate policy.

Computer Usage

Computer usage deals with how users use the computer systems you manage. An Acceptable Usage Policy (AUP) should define usage. An AUP describes appropriate and inappropriate computer system and network usage by users. Enforcing computer usage policies typically involves monitoring syslog log messages for user logons and logoffs, as well as monitoring the applications that end users use. One of the best methods of monitoring applications is to regularly run a program that logs which programs are running and notes the user account running the application. Over time you can build a database of which users typically use which applications. (This information can also assist in tasks such as performance tuning.)

User Awareness and Training

User awareness and training is the best preventative tool in your management arsenal as you try to ensure efficient and productive system use. As part of policy compliance monitoring, you need to enforce and regularly review proper training.

Conclusion

This chapter focused on the major areas of UNIX security. I developed a security policy that addressed the needs of the Operating Systems group, and I explained how to implement those needs in a UNIX environment. Throughout the chapter, I discussed several elements of incident response and disaster recovery, and I dedicated an entire section to each of these topics. Finally, I described the importance of policy compliance monitoring as it relates to users.