42.4. Pluggable Authentication Modules (PAM)

Programs that grant users access to a system use *authentication* to verify each other's identity (that is, to establish that a user is who they say they are).

Historically, each program had its own way of authenticating users. In Red Hat Enterprise Linux, many programs are configured to use a centralized authentication mechanism called *Pluggable Authentication Modules* (PAM).

PAM uses a pluggable, modular architecture, which affords the system administrator a great deal of flexibility in setting authentication policies for the system.

In most situations, the default PAM configuration file for a PAM-aware application is sufficient. Sometimes, however, it is necessary to edit a PAM configuration file. Because misconfiguration of PAM can compromise system security, it is important to understand the structure of these files before making any modifications. Refer to Section 42.4.3, “PAM Configuration File Format” for more information.

42.4.1. Advantages of PAM

PAM offers the following advantages:

- a common authentication scheme that can be used with a wide variety of applications.
- significant flexibility and control over authentication for both system administrators and application developers.
- a single, fully-documented library which allows developers to write programs without having to create their own authentication schemes.

42.4.2. PAM Configuration Files

The `/etc/pam.d/` directory contains the PAM configuration files for each PAM-aware application. In earlier versions of PAM, the `/etc/pam.conf` file was used, but this file is now deprecated and is only used if the `/etc/pam.d/` directory does not exist.

42.4.2.1. PAM Service Files

Each PAM-aware application or *service* has a file in the `/etc/pam.d/` directory. Each file in this directory has the same name as the service to which it controls access.

The PAM-aware program is responsible for defining its service name and installing its own PAM configuration file in the `/etc/pam.d/` directory. For example, the `login` program defines its service name as `login` and installs the `/etc/pam.d/login` PAM configuration file.

42.4.3. PAM Configuration File Format

Each PAM configuration file contains a group of directives formatted as follows:

```
<module interface>  <control flag>   <module name>   <module arguments>
```

Each of these elements is explained in the following sections.

42.4.3.1. Module Interface

Four types of PAM module interface are currently available. Each of these corresponds to a different aspect of the authorization process:

- `auth` — This module interface authenticates use. For example, it requests and verifies the validity of a password. Modules with this interface can also set credentials, such as group memberships or Kerberos tickets.
- `account` — This module interface verifies that access is allowed. For example, it may check if a user account has expired or if a user is allowed to log in at a particular time of day.
- `password` — This module interface is used for changing user passwords.
- `session` — This module interface configures and manages user sessions. Modules with this interface can also perform additional tasks that are needed to allow access, like mounting a user's home directory and making the user's mailbox available.

Note

An individual module can provide any or all module interfaces. For instance, `pam_unix.so` provides all four module interfaces.

In a PAM configuration file, the module interface is the first field defined. For example, a typical line in a configuration may look like this:

```
auth    required        pam_unix.so
```

This instructs PAM to use the `pam_unix.so` module's `auth` interface.

42.4.3.1.1. Stacking Module Interfaces

Module interface directives can be *stacked*, or placed upon one another, so that multiple modules are used together for one purpose. If a module's control flag uses the "sufficient" or "requisite" value (refer to Section 42.4.3.2, “Control Flag” for more information on these flags), then the order in which the modules are listed is important to the authentication process.

Stacking makes it easy for an administrator to require specific conditions to exist before allowing the user to authenticate. For example, the `reboot` command normally uses several stacked modules, as seen in its PAM configuration file:

```
[root@MyServer ~]# cat /etc/pam.d/reboot
#%PAM-1.0
auth    sufficient      pam_rootok.so
auth    required        pam_console.so
#auth   include system-auth
account required        pam_permit.so
```

- The first line is a comment and is not processed.

- `auth sufficient pam_rootok.so` — This line uses the `pam_rootok.so` module to check whether the current user is root, by verifying that their UID is 0. If this test succeeds, no other modules are consulted and the command is executed. If this test fails, the next module is consulted.
- `auth required pam_console.so` — This line uses the `pam_console.so` module to attempt to authenticate the user. If this user is already logged in at the console, `pam_console.so` checks whether there is a file in the `/etc/security/console.apps/` directory with the same name as the service name (reboot). If such a file exists, authentication succeeds and control is passed to the next module.
- `#auth include system-auth` — This line is commented and is not processed.
- `account required pam_permit.so` — This line uses the `pam_permit.so` module to allow the root user or anyone logged in at the console to reboot the system.

42.4.3.2. Control Flag

All PAM modules generate a success or failure result when called. Control flags tell PAM what do with the result. Modules can be stacked in a particular order, and the control flags determine how important the success or failure of a particular module is to the overall goal of authenticating the user to the service.

There are four predefined control flags:

- `required` — The module result must be successful for authentication to continue. If the test fails at this point, the user is not notified until the results of all module tests that reference that interface are complete.
- `requisite` — The module result must be successful for authentication to continue. However, if a test fails at this point, the user is notified immediately with a message reflecting the first failed `required` *or* `requisite` module test.
- `sufficient` — The module result is ignored if it fails. However, if the result of a module flagged `sufficient` is successful *and* no previous modules flagged `required` have failed, then no other results are required and the user is authenticated to the service.
- `optional` — The module result is ignored. A module flagged as `optional` only becomes necessary for successful authentication when no other modules reference the interface.

Important

The order in which `required` modules are called is not critical. Only the `sufficient` and `requisite` control flags cause order to become important.

A newer control flag syntax that allows for more precise control is now available for PAM.

The `pam.d` man page, and the PAM documentation, located in the `/usr/share/doc/pam-<version-number>/` directory, where *`<version-number>`* is the version number for PAM on your system, describe this newer syntax in detail.

42.4.3.3. Module Name

The module name provides PAM with the name of the pluggable module containing the specified module interface. In older versions of Red Hat Enterprise Linux, the full path to the module was provided in the PAM configuration file. However, since the advent of *multilib* systems, which store 64-bit PAM modules in the `/lib64/security/` directory, the directory name is omitted because the application is linked to the appropriate version of `libpam`, which can locate the correct version of the module.

42.4.3.4. Module Arguments

PAM uses *arguments* to pass information to a pluggable module during authentication for some modules.

For example, the `pam_userdb.so` module uses information stored in a Berkeley DB file to authenticate the user. Berkeley DB is an open source database system embedded in many applications. The module takes a `db` argument so that Berkeley DB knows which database to use for the requested service.

The following is a typical `pam_userdb.so` line in a PAM configuration. The *`<path-to-file>`* is the full path to the Berkeley DB database file:

```
auth    required        pam_userdb.so db=<path-to-file>
```

Invalid arguments are *generally* ignored and do not otherwise affect the success or failure of the PAM module. Some modules, however, may fail on invalid arguments. Most modules report errors to the `/var/log/secure` file.

42.4.4. Sample PAM Configuration Files

The following is a sample PAM application configuration file:

```
#%PAM-1.0
auth    required  pam_securetty.so
auth    required  pam_unix.so nullok
auth    required  pam_nologin.so
account required  pam_unix.so
password        required  pam_cracklib.so retry=3
password        required  pam_unix.so shadow nullok use_authtok
session required  pam_unix.so
```

- The first line is a comment, indicated by the hash mark (`#`) at the beginning of the line.
- Lines two through four stack three modules for login authentication.

  `auth required pam_securetty.so` — This module ensures that *if* the user is trying to log in as root, the tty on which the user is logging in is listed in the `/etc/securetty` file, *if* that file exists.

  If the tty is not listed in the file, any attempt to log in as root fails with a `Login incorrect` message.

`auth required pam_unix.so nullok` — This module prompts the user for a password and then checks the password using the information stored in `/etc/passwd` and, if it exists, `/etc/shadow`.

In the authentication phase, the `pam_unix.so` module automatically detects whether the user's password is in the `passwd` file or the `shadow` file. Refer to Section 32.6, "Shadow Passwords" for more information.

  - The argument `nullok` instructs the `pam_unix.so` module to allow a blank password.

- `auth required pam_nologin.so` — This is the final authentication step. It checks whether the `/etc/nologin` file exists. If it exists and the user is not root, authentication fails.

Note

In this example, all three `auth` modules are checked, even if the first `auth` module fails. This prevents the user from knowing at what stage their authentication failed. Such knowledge in the hands of an attacker could allow them to more easily deduce how to crack the system.

- `account required pam_unix.so` — This module performs any necessary account verification. For example, if shadow passwords have been enabled, the account interface of the `pam_unix.so` module checks to see if the account has expired or if the user has not changed the password within the allowed grace period.
- `password required pam_cracklib.so retry=3` — If a password has expired, the password component of the `pam_cracklib.so` module prompts for a new password. It then tests the newly created password to see whether it can easily be determined by a dictionary-based password cracking program.
  - The argument `retry=3` specifies that if the test fails the first time, the user has two more chances to create a strong password.
- `password required pam_unix.so shadow nullok use_authtok` — This line specifies that if the program changes the user's password, it should use the `password` interface of the `pam_unix.so` module to do so.
  - The argument `shadow` instructs the module to create shadow passwords when updating a user's password.
  - The argument `nullok` instructs the module to allow the user to change their password *from* a blank password, otherwise a null password is treated as an account lock.
  - The final argument on this line, `use_authtok`, provides a good example of the importance of order when stacking PAM modules. This argument instructs the module not to prompt the user for a new password. Instead, it accepts any password that was recorded by a previous password module. In this way, all new passwords must pass the `pam_cracklib.so` test for secure passwords before being accepted.
- `session required pam_unix.so` — The final line instructs the session interface of the `pam_unix.so` module to manage the session. This module logs the user name and the service type to `/var/log/secure` at the beginning and end of each session. This

module can be supplemented by stacking it with other session modules for additional functionality.

42.4.5. Creating PAM Modules

You can create or add new PAM modules at any time for use by PAM-aware applications.

For example, a developer might create a one-time-password creation method and write a PAM module to support it. PAM-aware programs can immediately use the new module and password method without being recompiled or otherwise modified.

This allows developers and system administrators to mix-and-match, as well as test, authentication methods for different programs without recompiling them.

Documentation on writing modules is included in the `/usr/share/doc/pam-<version-number>/` directory, where `<version-number>` is the version number for PAM on your system.

42.4.6. PAM and Administrative Credential Caching

A number of graphical administrative tools in Red Hat Enterprise Linux provide users with elevated privileges for up to five minutes using the `pam_timestamp.so` module. It is important to understand how this mechanism works, because a user who walks away from a terminal while `pam_timestamp.so` is in effect leaves the machine open to manipulation by anyone with physical access to the console.

In the PAM timestamp scheme, the graphical administrative application prompts the user for the root password when it is launched. When the user has been authenticated, the `pam_timestamp.so` module creates a timestamp file. By default, this is created in the `/var/run/sudo/` directory. If the timestamp file already exists, graphical administrative programs do not prompt for a password. Instead, the `pam_timestamp.so` module freshens the timestamp file, reserving an extra five minutes of unchallenged administrative access for the user.

You can verify the actual state of the timestamp file by inspecting the `/var/run/sudo/<user>` file. For the desktop, the relevant file is `unknown:root`. If it is present and its timestamp is less than five minutes old, the credentials are valid.

The existence of the timestamp file is indicated by an authentication icon, which appears in the notification area of the panel.

Figure 42.7. The Authentication Icon

42.4.6.1. Removing the Timestamp File

Before abandoning a console where a PAM timestamp is active, it is recommended that the timestamp file be destroyed. To do this from a graphical environment, click the

authentication icon on the panel. This causes a dialog box to appear. Click the **Forget Authorization** button to destroy the active timestamp file.

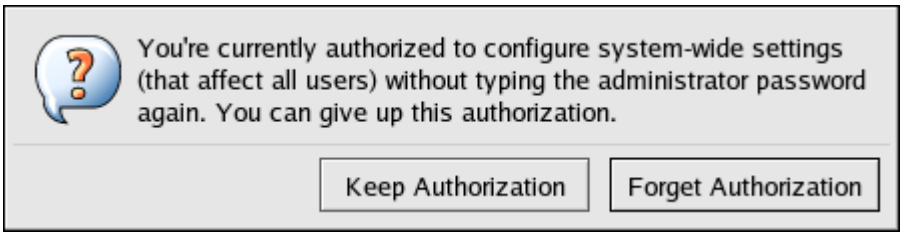


Figure 42.8. Dismiss Authentication Dialog

You should be aware of the following with respect to the PAM timestamp file:

- If logged in to the system remotely using `ssh`, use the `/sbin/pam_timestamp_check -k root` command to destroy the timestamp file.
- You need to run the `/sbin/pam_timestamp_check -k root` command from the same terminal window from which you launched the privileged application.
- You must be logged in as the user who originally invoked the `pam_timestamp.so` module in order to use the `/sbin/pam_timestamp_check -k` command. Do not log in as root to use this command.
- If you want to kill the credentials on the desktop (without using the **Forget Authorization** action on the icon), use the following command:
- `/sbin/pam_timestamp_check -k root </dev/null >/dev/null 2>/dev/null`

  Failure to use this command will only remove the credentials (if any) from the pty where you run the command.

Refer to the `pam_timestamp_check` man page for more information about destroying the timestamp file using `pam_timestamp_check`.

42.4.6.2. Common pam_timestamp Directives

The `pam_timestamp.so` module accepts several directives. The following are the two most commonly used options:

- `timestamp_timeout` — Specifies the period (in seconds) for which the timestamp file is valid. The default value is 300 (five minutes).
- `timestampdir` — Specifies the directory in which the timestamp file is stored. The default value is `/var/run/sudo/`.

Refer to Section 42.4.8.1, “Installed Documentation” for more information about controlling the `pam_timestamp.so` module.

42.4.7. PAM and Device Ownership

In Red Hat Enterprise Linux, the first user who logs in at the physical console of the machine can manipulate certain devices and perform certain tasks normally reserved for the root user. This is controlled by a PAM module called `pam_console.so`.

42.4.7.1. Device Ownership

When a user logs in to a Red Hat Enterprise Linux system, the `pam_console.so` module is called by `login` or the graphical login programs, **gdm**, **kdm**, and **xdm**. If this user is the first user to log in at the physical console — referred to as the *console user* — the module grants the user ownership of a variety of devices normally owned by root. The console user owns these devices until the last local session for that user ends. After this user has logged out, ownership of the devices reverts back to the root user.

The devices affected include, but are not limited to, sound cards, diskette drives, and CD-ROM drives.

This facility allows a local user to manipulate these devices without obtaining root access, thus simplifying common tasks for the console user.

You can modify the list of devices controlled by `pam_console.so` by editing the following files:

- `/etc/security/console.perms`
- `/etc/security/console.perms.d/50-default.perms`

You can change the permissions of different devices than those listed in the above files, or override the specified defaults. Rather than modify the `50-default.perms` file, you should create a new file (for example, *xx*`-name.perms`) and enter the required modifications. The name of the new default file must begin with a number higher than 50 (for example, `51-default.perms`). This will override the defaults in the `50-default.perms` file.

Warning

If the **gdm**, **kdm**, or **xdm** display manager configuration file has been altered to allow remote users to log in *and* the host is configured to run at runlevel 5, it is advisable to change the `<console>` and `<xconsole>` directives in the `/etc/security/console.perms` to the following values:

```
<console>=tty[0-9][0-9]* vc/[0-9][0-9]* :0\.[0-9] :0
<xconsole>=:0\.[0-9] :0
```

This prevents remote users from gaining access to devices and restricted applications on the machine.

If the **gdm**, **kdm**, or **xdm** display manager configuration file has been altered to allow remote users to log in *and* the host is configured to run at any multiple user runlevel other than 5, it is advisable to remove the `<xconsole>` directive entirely and change the `<console>` directive to the following value:

```
<console>=tty[0-9][0-9]* vc/[0-9][0-9]*
```

42.4.7.2. Application Access

The console user also has access to certain programs configured for use in the `/etc/security/console.apps/` directory.

This directory contains configuration files which enable the console user to run certain applications in `/sbin` and `/usr/sbin`.

These configuration files have the same name as the applications that they set up.

One notable group of applications that the console user has access to are three programs that shut down or reboot the system:

- `/sbin/halt`
- `/sbin/reboot`
- `/sbin/poweroff`

Because these are PAM-aware applications, they call the `pam_console.so` module as a requirement for use.

Refer to Section 42.4.8.1, “Installed Documentation” for more information.

42.4.8. Additional Resources

The following resources further explain methods to use and configure PAM. In addition to these resources, read the PAM configuration files on the system to better understand how they are structured.

42.4.8.1. Installed Documentation

- PAM-related man pages — Several man pages exist for the various applications and configuration files involved with PAM. The following is a list of some of the more important man pages.

  Configuration Files

  - `pam` — Good introductory information on PAM, including the structure and purpose of the PAM configuration files.

    Note that this man page discusses both `/etc/pam.conf` and individual configuration files in the `/etc/pam.d/` directory. By default, Red Hat Enterprise Linux uses the individual configuration files in the `/etc/pam.d/` directory, ignoring `/etc/pam.conf` even if it exists.

  - `pam_console` — Describes the purpose of the `pam_console.so` module. It also describes the appropriate syntax for an entry within a PAM configuration file.
  - `console.apps` — Describes the format and options available in the `/etc/security/console.apps` configuration file, which defines which applications are accessible by the console user assigned by PAM.

    - `console.perms` — Describes the format and options available in the `/etc/security/console.perms` configuration file, which specifies the console user permissions assigned by PAM.
    - `pam_timestamp` — Describes the `pam_timestamp.so` module.
- `/usr/share/doc/pam-<version-number>` — Contains a *System Administrators' Guide*, a *Module Writers' Manual*, and the *Application Developers' Manual*, as well as a copy of the PAM standard, DCE-RFC 86.0, where `<version-number>` is the version number of PAM.
- `/usr/share/doc/pam-<version-number>/txts/README.pam_timestamp` — Contains information about the `pam_timestamp.so` PAM module, where `<version-number>` is the version number of PAM.

42.4.8.2. Useful Websites

- http://www.kernel.org/pub/linux/libs/pam/ — The primary distribution website for the Linux-PAM project, containing information on various PAM modules, a FAQ, and additional PAM documentation.

> **Note**
>
> The documentation in the above website is for the last released upstream version of PAM and might not be 100% accurate for the PAM version included in Red Hat Enterprise Linux.

Chapter 41. Security Overview

Because of the increased reliance on powerful, networked computers to help run businesses and keep track of our personal information, industries have been formed around the practice of network and computer security. Enterprises have solicited the knowledge and skills of security experts to properly audit systems and tailor solutions to fit the operating requirements of the organization. Because most organizations are dynamic in nature, with workers accessing company IT resources locally and remotely, the need for secure computing environments has become more pronounced.

Unfortunately, most organizations (as well as individual users) regard security as an afterthought, a process that is overlooked in favor of increased power, productivity, and budgetary concerns. Proper security implementation is often enacted *postmortem* — after an unauthorized intrusion has already occurred. Security experts agree that the right measures taken prior to connecting a site to an untrusted network, such as the Internet, is an effective means of thwarting most attempts at intrusion.

41.1. Introduction to Security

41.1.1. What is Computer Security?

Computer security is a general term that covers a wide area of computing and information processing. Industries that depend on computer systems and networks to conduct daily business transactions and access crucial information regard their data as an important part of their overall assets. Several terms and metrics have entered our daily business vocabulary, such as total cost of ownership (TCO) and quality of service (QoS). In these metrics, industries calculate aspects such as data integrity and high-availability as part of their planning and process management costs. In some industries, such as electronic commerce, the availability and trustworthiness of data can be the difference between success and failure.

41.1.1.1. How did Computer Security Come about?

Information security has evolved over the years due to the increasing reliance on public networks not to disclose personal, financial, and other restricted information. There are numerous instances such as the Mitnick and the Vladimir Levin cases that prompted organizations across all industries to rethink the way they handle information transmission and disclosure. The popularity of the Internet was one of the most important developments that prompted an intensified effort in data security.

An ever-growing number of people are using their personal computers to gain access to the resources that the Internet has to offer. From research and information retrieval to electronic mail and commerce transaction, the Internet has been regarded as one of the most important developments of the 20th century.

The Internet and its earlier protocols, however, were developed as a *trust-based* system. That is, the Internet Protocol was not designed to be secure in itself. There are no approved security standards built into the TCP/IP communications stack, leaving it open to potentially malicious users and processes across the network. Modern developments have made Internet communication more secure, but there are still several incidents that gain national attention and alert us to the fact that nothing is completely safe.

41.1.1.2. Security Today

In February of 2000, a Distributed Denial of Service (DDoS) attack was unleashed on several of the most heavily-trafficked sites on the Internet. The attack rendered yahoo.com, cnn.com, amazon.com, fbi.gov, and several other sites completely unreachable to normal users, as it tied up routers for several hours with large-byte ICMP packet transfers, also called a *ping flood*. The attack was brought on by unknown assailants using specially created, widely available programs that scanned vulnerable network servers, installed client applications called *trojans* on the servers, and timed an attack with every infected server flooding the victim sites and rendering them unavailable. Many blame the attack on fundamental flaws in the way routers and the protocols used are structured to accept all incoming data, no matter where or for what purpose the packets are sent.

Currently, an estimated 945 million people use or have used the Internet worldwide (Computer Industry Almanac, 2004). At the same time:

- On any given day, there are approximately 225 major incidences of security breach reported to the CERT Coordination Center at Carnegie Mellon University.[11]

- In 2003, the number of CERT reported incidences jumped to 137,529 from 82,094 in 2002 and from 52,658 in 2001.[12]
- The worldwide economic impact of the three most dangerous Internet Viruses of the last three years was estimated at US$13.2 Billion.[13]

Computer security has become a quantifiable and justifiable expense for all IT budgets. Organizations that require data integrity and high availability elicit the skills of system administrators, developers, and engineers to ensure 24x7 reliability of their systems, services, and information. Falling victim to malicious users, processes, or coordinated attacks is a direct threat to the success of the organization.

Unfortunately, system and network security can be a difficult proposition, requiring an intricate knowledge of how an organization regards, uses, manipulates, and transmits its information. Understanding the way an organization (and the people that make up the organization) conducts business is paramount to implementing a proper security plan.

41.1.1.3. Standardizing Security

Enterprises in every industry rely on regulations and rules that are set by standards making bodies such as the American Medical Association (AMA) or the Institute of Electrical and Electronics Engineers (IEEE). The same ideals hold true for information security. Many security consultants and vendors agree upon the standard security model known as CIA, or *Confidentiality, Integrity, and Availability*. This three-tiered model is a generally accepted component to assessing risks of sensitive information and establishing security policy. The following describes the CIA model in further detail:

- Confidentiality — Sensitive information must be available only to a set of pre-defined individuals. Unauthorized transmission and usage of information should be restricted. For example, confidentiality of information ensures that a customer's personal or financial information is not obtained by an unauthorized individual for malicious purposes such as identity theft or credit fraud.
- Integrity — Information should not be altered in ways that render it incomplete or incorrect. Unauthorized users should be restricted from the ability to modify or destroy sensitive information.
- Availability — Information should be accessible to authorized users any time that it is needed. Availability is a warranty that information can be obtained with an agreed-upon frequency and timeliness. This is often measured in terms of percentages and agreed to formally in Service Level Agreements (SLAs) used by network service providers and their enterprise clients.

41.1.2. Security Controls

Computer security is often divided into three distinct master categories, commonly referred to as *controls*:

- Physical
- Technical
- Administrative

These three broad categories define the main objectives of proper security implementation. Within these controls are sub-categories that further detail the controls and how to implement them.

41.1.2.1. Physical Controls

Physical control is the implementation of security measures in a defined structure used to deter or prevent unauthorized access to sensitive material. Examples of physical controls are:

- Closed-circuit surveillance cameras
- Motion or thermal alarm systems
- Security guards
- Picture IDs
- Locked and dead-bolted steel doors
- Biometrics (includes fingerprint, voice, face, iris, handwriting, and other automated methods used to recognize individuals)

41.1.2.2. Technical Controls

Technical controls use technology as a basis for controlling the access and usage of sensitive data throughout a physical structure and over a network. Technical controls are far-reaching in scope and encompass such technologies as:

- Encryption
- Smart cards
- Network authentication
- Access control lists (ACLs)
- File integrity auditing software

41.1.2.3. Administrative Controls

Administrative controls define the human factors of security. It involves all levels of personnel within an organization and determines which users have access to what resources and information by such means as:

- Training and awareness
- Disaster preparedness and recovery plans
- Personnel recruitment and separation strategies
- Personnel registration and accounting

41.1.3. Conclusion

Now that you have learned about the origins, reasons, and aspects of security, you can determine the appropriate course of action with regard to Red Hat Enterprise Linux. It is important to know what factors and conditions make up security in order to plan and implement a proper strategy. With this information in mind, the process can be formalized and the path becomes clearer as you delve deeper into the specifics of the security process.

41.2. Vulnerability Assessment

Given time, resources, and motivation, a cracker can break into nearly any system. At the end of the day, all of the security procedures and technologies currently available cannot guarantee that any systems are safe from intrusion. Routers help secure gateways to the Internet. Firewalls help secure the edge of the network. Virtual Private Networks safely pass data in an encrypted stream. Intrusion detection systems warn you of malicious activity. However, the success of each of these technologies is dependent upon a number of variables, including:

- The expertise of the staff responsible for configuring, monitoring, and maintaining the technologies.
- The ability to patch and update services and kernels quickly and efficiently.
- The ability of those responsible to keep constant vigilance over the network.

Given the dynamic state of data systems and technologies, securing corporate resources can be quite complex. Due to this complexity, it is often difficult to find expert resources for all of your systems. While it is possible to have personnel knowledgeable in many areas of information security at a high level, it is difficult to retain staff who are experts in more than a few subject areas. This is mainly because each subject area of information security requires constant attention and focus. Information security does not stand still.

41.2.1. Thinking Like the Enemy

Suppose that you administer an enterprise network. Such networks are commonly comprised of operating systems, applications, servers, network monitors, firewalls, intrusion detection systems, and more. Now imagine trying to keep current with each of these. Given the complexity of today's software and networking environments, exploits and bugs are a certainty. Keeping current with patches and updates for an entire network can prove to be a daunting task in a large organization with heterogeneous systems.

Combine the expertise requirements with the task of keeping current, and it is inevitable that adverse incidents occur, systems are breached, data is corrupted, and service is interrupted.

To augment security technologies and aid in protecting systems, networks, and data, you must think like a cracker and gauge the security of your systems by checking for weaknesses. Preventative vulnerability assessments against your own systems and network resources can reveal potential issues that can be addressed before a cracker exploits it.

A vulnerability assessment is an internal audit of your network and system security; the results of which indicate the confidentiality, integrity, and availability of your network (as explained in Section 41.1.1.3, “Standardizing Security”). Typically, vulnerability assessment starts with a reconnaissance phase, during which important data regarding the target systems and resources is gathered. This phase leads to the system readiness phase, whereby the target is essentially checked for all known vulnerabilities. The readiness phase culminates in the reporting phase, where the findings are classified into categories of high, medium, and low risk; and methods for improving the security (or mitigating the risk of vulnerability) of the target are discussed.

If you were to perform a vulnerability assessment of your home, you would likely check each door to your home to see if they are closed and locked. You would also check every window, making sure that they closed completely and latch correctly. This same concept applies to

systems, networks, and electronic data. Malicious users are the thieves and vandals of your data. Focus on their tools, mentality, and motivations, and you can then react swiftly to their actions.

41.2.2. Defining Assessment and Testing

Vulnerability assessments may be broken down into one of two types: *Outside looking in* and *inside looking around*.

When performing an outside looking in vulnerability assessment, you are attempting to compromise your systems from the outside. Being external to your company provides you with the cracker's viewpoint. You see what a cracker sees — publicly-routable IP addresses, systems on your *DMZ*, external interfaces of your firewall, and more. DMZ stands for "demilitarized zone", which corresponds to a computer or small subnetwork that sits between a trusted internal network, such as a corporate private LAN, and an untrusted external network, such as the public Internet. Typically, the DMZ contains devices accessible to Internet traffic, such as Web (HTTP) servers, FTP servers, SMTP (e-mail) servers and DNS servers.

When you perform an inside looking around vulnerability assessment, you are somewhat at an advantage since you are internal and your status is elevated to trusted. This is the viewpoint you and your co-workers have once logged on to your systems. You see print servers, file servers, databases, and other resources.

There are striking distinctions between these two types of vulnerability assessments. Being internal to your company gives you elevated privileges more so than any outsider. Still today in most organizations, security is configured in such a manner as to keep intruders out. Very little is done to secure the internals of the organization (such as departmental firewalls, user-level access controls, authentication procedures for internal resources, and more). Typically, there are many more resources when looking around inside as most systems are internal to a company. Once you set yourself outside of the company, you immediately are given an untrusted status. The systems and resources available to you externally are usually very limited.

Consider the difference between vulnerability assessments and *penetration tests*. Think of a vulnerability assessment as the first step to a penetration test. The information gleaned from the assessment is used for testing. Whereas, the assessment is checking for holes and potential vulnerabilities, the penetration testing actually attempts to exploit the findings.

Assessing network infrastructure is a dynamic process. Security, both information and physical, is dynamic. Performing an assessment shows an overview, which can turn up false positives and false negatives.

Security administrators are only as good as the tools they use and the knowledge they retain. Take any of the assessment tools currently available, run them against your system, and it is almost a guarantee that there are some false positives. Whether by program fault or user error, the result is the same. The tool may find vulnerabilities which in reality do not exist (false positive); or, even worse, the tool may not find vulnerabilities that actually do exist (false negative).

Now that the difference between a vulnerability assessment and a penetration test is defined, take the findings of the assessment and review them carefully before conducting a penetration test as part of your new best practices approach.

Warning

Attempting to exploit vulnerabilities on production resources can have adverse effects to the productivity and efficiency of your systems and network.

The following list examines some of the benefits to performing vulnerability assessments.

- Creates proactive focus on information security
- Finds potential exploits before crackers find them
- Results in systems being kept up to date and patched
- Promotes growth and aids in developing staff expertise
- Abates Financial loss and negative publicity

41.2.2.1. Establishing a Methodology

To aid in the selection of tools for a vulnerability assessment, it is helpful to establish a vulnerability assessment methodology. Unfortunately, there is no predefined or industry approved methodology at this time; however, common sense and best practices can act as a sufficient guide.

What is the target? Are we looking at one server, or are we looking at our entire network and everything within the network? Are we external or internal to the company? The answers to these questions are important as they help determine not only which tools to select but also the manner in which they are used.

To learn more about establishing methodologies, refer to the following websites:

- http://www.isecom.org/projects/osstmm.htm *The Open Source Security Testing Methodology Manual* (OSSTMM)
- http://www.owasp.org/ *The Open Web Application Security Project*

41.2.3. Evaluating the Tools

An assessment can start by using some form of an information gathering tool. When assessing the entire network, map the layout first to find the hosts that are running. Once located, examine each host individually. Focusing on these hosts requires another set of tools. Knowing which tools to use may be the most crucial step in finding vulnerabilities.

Just as in any aspect of everyday life, there are many different tools that perform the same job. This concept applies to performing vulnerability assessments as well. There are tools specific to operating systems, applications, and even networks (based on the protocols used). Some tools are free; others are not. Some tools are intuitive and easy to use, while others are cryptic and poorly documented but have features that other tools do not.

Finding the right tools may be a daunting task and in the end, experience counts. If possible, set up a test lab and try out as many tools as you can, noting the strengths and weaknesses of each. Review the README file or man page for the tool. Additionally, look to the Internet for more information, such as articles, step-by-step guides, or even mailing lists specific to a tool.

The tools discussed below are just a small sampling of the available tools.

41.2.3.1. Scanning Hosts with Nmap

Nmap is a popular tool included in Red Hat Enterprise Linux that can be used to determine the layout of a network. Nmap has been available for many years and is probably the most often used tool when gathering information. An excellent man page is included that provides a detailed description of its options and usage. Administrators can use Nmap on a network to find host systems and open ports on those systems.

Nmap is a competent first step in vulnerability assessment. You can map out all the hosts within your network and even pass an option that allows Nmap to attempt to identify the operating system running on a particular host. Nmap is a good foundation for establishing a policy of using secure services and stopping unused services.

41.2.3.1.1. Using Nmap

Nmap can be run from a shell prompt by typing the `nmap` command followed by the hostname or IP address of the machine to scan.

```
nmap foo.example.com
```

The results of the scan (which could take up to a few minutes, depending on where the host is located) should look similar to the following:

```
Starting nmap V. 3.50 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1591 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
111/tcp    open        sunrpc
443/tcp    open        https
515/tcp    open        printer
950/tcp    open        oftep-rpc
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 71.825 seconds
```

Nmap tests the most common network communication ports for listening or waiting services. This knowledge can be helpful to an administrator who wants to close down unnecessary or unused services.

For more information about using Nmap, refer to the official homepage at the following URL:

http://www.insecure.org/

41.2.3.2. Nessus

Nessus is a full-service security scanner. The plug-in architecture of Nessus allows users to customize it for their systems and networks. As with any scanner, Nessus is only as good as the signature database it relies upon. Fortunately, Nessus is frequently updated and features full reporting, host scanning, and real-time vulnerability searches. Remember that there could be false positives and false negatives, even in a tool as powerful and as frequently updated as Nessus.

Note

Nessus is not included with Red Hat Enterprise Linux and is not supported. It has been included in this document as a reference to users who may be interested in using this popular application.

For more information about Nessus, refer to the official website at the following URL:

http://www.nessus.org/

41.2.3.3. Nikto

Nikto is an excellent common gateway interface (CGI) script scanner. Nikto not only checks for CGI vulnerabilities but does so in an evasive manner, so as to elude intrusion detection systems. It comes with thorough documentation which should be carefully reviewed prior to running the program. If you have Web servers serving up CGI scripts, Nikto can be an excellent resource for checking the security of these servers.

Note

Nikto is not included with Red Hat Enterprise Linux and is not supported. It has been included in this document as a reference to users who may be interested in using this popular application.

More information about Nikto can be found at the following URL:

http://www.cirt.net/code/nikto.shtml

41.2.3.4. VLAD the Scanner

VLAD is a vulnerabilities scanner developed by the RAZOR team at Bindview, Inc., which checks for the SANS Top Ten list of common security issues (SNMP issues, file sharing issues, etc.). While not as full-featured as Nessus, VLAD is worth investigating.

Note

VLAD is not included with Red Hat Enterprise Linux and is not supported. It has been included in this document as a reference to users who may be interested in using this popular application.

More information about VLAD can be found on the RAZOR team website at the following URL:

http://www.bindview.com/Support/Razor/Utilities/

41.2.3.5. Anticipating Your Future Needs

Depending upon your target and resources, there are many tools available. There are tools for wireless networks, Novell networks, Windows systems, Linux systems, and more. Another essential part of performing assessments may include reviewing physical security, personnel screening, or voice/PBX network assessment. New concepts, such as *war walking* scanning the perimeter of your enterprise's physical structures for wireless network vulnerabilities are some emerging concepts that you can investigate and, if needed, incorporate into your assessments. Imagination and exposure are the only limits of planning and conducting vulnerability assessments.

41.3. Attackers and Vulnerabilities

To plan and implement a good security strategy, first be aware of some of the issues which determined, motivated attackers exploit to compromise systems. But before detailing these issues, the terminology used when identifying an attacker must be defined.

41.3.1. A Quick History of Hackers

The modern meaning of the term *hacker* has origins dating back to the 1960s and the Massachusetts Institute of Technology (MIT) Tech Model Railroad Club, which designed train sets of large scale and intricate detail. Hacker was a name used for club members who discovered a clever trick or workaround for a problem.

The term hacker has since come to describe everything from computer buffs to gifted programmers. A common trait among most hackers is a willingness to explore in detail how computer systems and networks function with little or no outside motivation. Open source software developers often consider themselves and their colleagues to be hackers, and use the word as a term of respect.

Typically, hackers follow a form of the *hacker ethic* which dictates that the quest for information and expertise is essential, and that sharing this knowledge is the hackers duty to the community. During this quest for knowledge, some hackers enjoy the academic challenges of circumventing security controls on computer systems. For this reason, the press often uses the term hacker to describe those who illicitly access systems and networks with unscrupulous, malicious, or criminal intent. The more accurate term for this type of computer hacker is *cracker* — a term created by hackers in the mid-1980s to differentiate the two communities.

41.3.1.1. Shades of Gray

Within the community of individuals who find and exploit vulnerabilities in systems and networks are several distinct groups. These groups are often described by the shade of hat

that they "wear" when performing their security investigations and this shade is indicative of their intent.

The *white hat hacker* is one who tests networks and systems to examine their performance and determine how vulnerable they are to intrusion. Usually, white hat hackers crack their own systems or the systems of a client who has specifically employed them for the purposes of security auditing. Academic researchers and professional security consultants are two examples of white hat hackers.

A *black hat hacker* is synonymous with a cracker. In general, crackers are less focused on programming and the academic side of breaking into systems. They often rely on available cracking programs and exploit well known vulnerabilities in systems to uncover sensitive information for personal gain or to inflict damage on the target system or network.

The *gray hat hacker*, on the other hand, has the skills and intent of a white hat hacker in most situations but uses his knowledge for less than noble purposes on occasion. A gray hat hacker can be thought of as a white hat hacker who wears a black hat at times to accomplish his own agenda.

Gray hat hackers typically subscribe to another form of the hacker ethic, which says it is acceptable to break into systems as long as the hacker does not commit theft or breach confidentiality. Some would argue, however, that the act of breaking into a system is in itself unethical.

Regardless of the intent of the intruder, it is important to know the weaknesses a cracker may likely attempt to exploit. The remainder of the chapter focuses on these issues.

41.3.2. Threats to Network Security

Bad practices when configuring the following aspects of a network can increase the risk of attack.

41.3.2.1. Insecure Architectures

A misconfigured network is a primary entry point for unauthorized users. Leaving a trust-based, open local network vulnerable to the highly-insecure Internet is much like leaving a door ajar in a crime-ridden neighborhood — nothing may happen for an arbitrary amount of time, but *eventually* someone exploits the opportunity.

41.3.2.1.1. Broadcast Networks

System administrators often fail to realize the importance of networking hardware in their security schemes. Simple hardware such as hubs and routers rely on the broadcast or non-switched principle; that is, whenever a node transmits data across the network to a recipient node, the hub or router sends a broadcast of the data packets until the recipient node receives and processes the data. This method is the most vulnerable to address resolution protocol (*arp*) or media access control (*MAC*) address spoofing by both outside intruders and unauthorized users on local hosts.

41.3.2.1.2. Centralized Servers

Another potential networking pitfall is the use of centralized computing. A common cost-cutting measure for many businesses is to consolidate all services to a single powerful machine. This can be convenient as it is easier to manage and costs considerably less than multiple-server configurations. However, a centralized server introduces a single point of failure on the network. If the central server is compromised, it may render the network completely useless or worse, prone to data manipulation or theft. In these situations, a central server becomes an open door which allows access to the entire network.

41.3.3. Threats to Server Security

Server security is as important as network security because servers often hold a great deal of an organization's vital information. If a server is compromised, all of its contents may become available for the cracker to steal or manipulate at will. The following sections detail some of the main issues.

41.3.3.1. Unused Services and Open Ports

A full installation of Red Hat Enterprise Linux contains 1000+ application and library packages. However, most server administrators do not opt to install every single package in the distribution, preferring instead to install a base installation of packages, including several server applications.

A common occurrence among system administrators is to install the operating system without paying attention to what programs are actually being installed. This can be problematic because unneeded services may be installed, configured with the default settings, and possibly turned on. This can cause unwanted services, such as Telnet, DHCP, or DNS, to run on a server or workstation without the administrator realizing it, which in turn can cause unwanted traffic to the server, or even, a potential pathway into the system for crackers. Refer To Section 42.2, “Server Security” for information on closing ports and disabling unused services.

41.3.3.2. Unpatched Services

Most server applications that are included in a default installation are solid, thoroughly tested pieces of software. Having been in use in production environments for many years, their code has been thoroughly refined and many of the bugs have been found and fixed.

However, there is no such thing as perfect software and there is always room for further refinement. Moreover, newer software is often not as rigorously tested as one might expect, because of its recent arrival to production environments or because it may not be as popular as other server software.

Developers and system administrators often find exploitable bugs in server applications and publish the information on bug tracking and security-related websites such as the Bugtraq mailing list (http://www.securityfocus.com) or the Computer Emergency Response Team (CERT) website (http://www.cert.org). Although these mechanisms are an effective way of alerting the community to security vulnerabilities, it is up to system administrators to patch their systems promptly. This is particularly true because crackers have access to these same vulnerability tracking services and will use the information to crack unpatched systems

whenever they can. Good system administration requires vigilance, constant bug tracking, and proper system maintenance to ensure a more secure computing environment.

Refer to Section 41.5, “Security Updates” for more information about keeping a system up-to-date.

41.3.3.3. Inattentive Administration

Administrators who fail to patch their systems are one of the greatest threats to server security. According to the *System Administration Network and Security Institute* (*SANS*), the primary cause of computer security vulnerability is to "assign untrained people to maintain security and provide neither the training nor the time to make it possible to do the job."[14] This applies as much to inexperienced administrators as it does to overconfident or amotivated administrators.

Some administrators fail to patch their servers and workstations, while others fail to watch log messages from the system kernel or network traffic. Another common error is when default passwords or keys to services are left unchanged. For example, some databases have default administration passwords because the database developers assume that the system administrator changes these passwords immediately after installation. If a database administrator fails to change this password, even an inexperienced cracker can use a widely-known default password to gain administrative privileges to the database. These are only a few examples of how inattentive administration can lead to compromised servers.

41.3.3.4. Inherently Insecure Services

Even the most vigilant organization can fall victim to vulnerabilities if the network services they choose are inherently insecure. For instance, there are many services developed under the assumption that they are used over trusted networks; however, this assumption fails as soon as the service becomes available over the Internet — which is itself inherently untrusted.

One category of insecure network services are those that require unencrypted usernames and passwords for authentication. Telnet and FTP are two such services. If packet sniffing software is monitoring traffic between the remote user and such a service usernames and passwords can be easily intercepted.

Inherently, such services can also more easily fall prey to what the security industry terms the *man-in-the-middle* attack. In this type of attack, a cracker redirects network traffic by tricking a cracked name server on the network to point to his machine instead of the intended server. Once someone opens a remote session to the server, the attacker's machine acts as an invisible conduit, sitting quietly between the remote service and the unsuspecting user capturing information. In this way a cracker can gather administrative passwords and raw data without the server or the user realizing it.

Another category of insecure services include network file systems and information services such as NFS or NIS, which are developed explicitly for LAN usage but are, unfortunately, extended to include WANs (for remote users). NFS does not, by default, have any authentication or security mechanisms configured to prevent a cracker from mounting the NFS share and accessing anything contained therein. NIS, as well, has vital information that

must be known by every computer on a network, including passwords and file permissions, within a plain text ASCII or DBM (ASCII-derived) database. A cracker who gains access to this database can then access every user account on a network, including the administrator's account.

By default, Red Hat Enterprise Linux is released with all such services turned off. However, since administrators often find themselves forced to use these services, careful configuration is critical. Refer to Section 42.2, “Server Security” for more information about setting up services in a safe manner.

41.3.4. Threats to Workstation and Home PC Security

Workstations and home PCs may not be as prone to attack as networks or servers, but since they often contain sensitive data, such as credit card information, they are targeted by system crackers. Workstations can also be co-opted without the user's knowledge and used by attackers as "slave" machines in coordinated attacks. For these reasons, knowing the vulnerabilities of a workstation can save users the headache of reinstalling the operating system, or worse, recovering from data theft.

41.3.4.1. Bad Passwords

Bad passwords are one of the easiest ways for an attacker to gain access to a system. For more on how to avoid common pitfalls when creating a password, refer to Section 42.1.3, “Password Security”.

41.3.4.2. Vulnerable Client Applications

Although an administrator may have a fully secure and patched server, that does not mean remote users are secure when accessing it. For instance, if the server offers Telnet or FTP services over a public network, an attacker can capture the plain text usernames and passwords as they pass over the network, and then use the account information to access the remote user's workstation.

Even when using secure protocols, such as SSH, a remote user may be vulnerable to certain attacks if they do not keep their client applications updated. For instance, v.1 SSH clients are vulnerable to an X-forwarding attack from malicious SSH servers. Once connected to the server, the attacker can quietly capture any keystrokes and mouse clicks made by the client over the network. This problem was fixed in the v.2 SSH protocol, but it is up to the user to keep track of what applications have such vulnerabilities and update them as necessary.

Section 42.1, “Workstation Security” discusses in more detail what steps administrators and home users should take to limit the vulnerability of computer workstations.

41.4. Common Exploits and Attacks

Table 41.1, “Common Exploits” details some of the most common exploits and entry points used by intruders to access organizational network resources. Key to these common exploits are the explanations of how they are performed and how administrators can properly safeguard their network against such attacks.

Exploit	Description	Notes
Null or Default Passwords	Leaving administrative passwords blank or using a default password set by the product vendor. This is most common in hardware such as routers and firewalls, though some services that run on Linux can contain default administrator passwords (though Red Hat Enterprise Linux 5 does not ship with them).	Commonly associated with networking hardware such as routers, firewalls, VPNs, and network attached storage (NAS) appliances. Common in many legacy operating systems, especially OSes that bundle services (such as UNIX and Windows.) Administrators sometimes create privileged user accounts in a rush and leave the password null, a perfect entry point for malicious users who discover the account.
Default Shared Keys	Secure services sometimes package default security keys for development or evaluation testing purposes. If these keys are left unchanged and are placed in a production environment on the Internet, *all* users with the same default keys have access to that shared-key resource, and any sensitive information that it contains.	Most common in wireless access points and preconfigured secure server appliances.
IP Spoofing	A remote machine acts as a node on your local network, finds vulnerabilities with your servers, and installs a backdoor program or trojan horse to gain control over your network resources.	Spoofing is quite difficult as it involves the attacker predicting TCP/IP SYN-ACK numbers to coordinate a connection to target systems, but several tools are available to assist crackers in performing such a vulnerability. Depends on target system running services (such as `rsh`, `telnet`, FTP and others) that use *source-based* authentication techniques, which are not recommended when compared to PKI or other forms of encrypted authentication used in `ssh` or SSL/TLS.
Eavesdropping	Collecting data that passes between two active nodes on a network by eavesdropping on the connection between the two nodes.	This type of attack works mostly with plain text transmission protocols such as Telnet, FTP, and HTTP transfers. Remote attacker must have access to a compromised system on a LAN in order to perform such an attack;

Exploit	Description	Notes
		usually the cracker has used an active attack (such as IP spoofing or man-in-the-middle) to compromise a system on the LAN. Preventative measures include services with cryptographic key exchange, one-time passwords, or encrypted authentication to prevent password snooping; strong encryption during transmission is also advised.
Service Vulnerabilities	An attacker finds a flaw or loophole in a service run over the Internet; through this vulnerability, the attacker compromises the entire system and any data that it may hold, and could possibly compromise other systems on the network.	HTTP-based services such as CGI are vulnerable to remote command execution and even interactive shell access. Even if the HTTP service runs as a non-privileged user such as "nobody", information such as configuration files and network maps can be read, or the attacker can start a denial of service attack which drains system resources or renders it unavailable to other users. Services sometimes can have vulnerabilities that go unnoticed during development and testing; these vulnerabilities (such as *buffer overflows*, where attackers crash a service using arbitary values that fill the memory buffer of an application, giving the attacker an interactive command prompt from which they may execute arbitrary commands) can give complete administrative control to an attacker. Administrators should make sure that services do not run as the root user, and should stay vigilant of patches and errata updates for applications from vendors or security organizations such as CERT and CVE.
Application Vulnerabilities	Attackers find faults in desktop and workstation applications (such as e-mail clients) and execute arbitrary code, implant trojan horses for future compromise, or crash	Workstations and desktops are more prone to exploitation as workers do not have the expertise or experience to prevent or detect a compromise; it is imperative to inform individuals of

Exploit	Description	Notes
	systems. Further exploitation can occur if the compromised workstation has administrative privileges on the rest of the network.	the risks they are taking when they install unauthorized software or open unsolicited email attachments. Safeguards can be implemented such that email client software does not automatically open or execute attachments. Additionally, the automatic update of workstation software via Red Hat Network or other system management services can alleviate the burdens of multi-seat security deployments.
Denial of Service (DoS) Attacks	Attacker or group of attackers coordinate against an organization's network or server resources by sending unauthorized packets to the target host (either server, router, or workstation). This forces the resource to become unavailable to legitimate users.	The most reported DoS case in the US occurred in 2000. Several highly-trafficked commercial and government sites were rendered unavailable by a coordinated ping flood attack using several compromised systems with high bandwidth connections acting as *zombies*, or redirected broadcast nodes. Source packets are usually forged (as well as rebroadcasted), making investigation as to the true source of the attack difficult. Advances in ingress filtering (IETF rfc2267) using `iptables` and Network IDSes such as `snort` assist administrators in tracking down and preventing distributed DoS attacks.

Types of permissions

Although there are already a lot of good security features built into Linux-based systems, one very important potential vulnerability can exist when local access is granted - - that is file permission based issues resulting from a user not assigning the correct permissions to files and directories. So based upon the need for proper permissions, I will go over the ways to assign permissions and show you some examples where modification may be necessary.

Basic File Permissions

Permission Groups

Each file and directory has three user based permission groups:

- **owner** - The Owner permissions apply only the owner of the file or directory, they will not impact the actions of other users.
- **group** - The Group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.

- **all users** - The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

Permission Types

Each file or directory has three basic permission types:

- **read** - The Read permission refers to a user's capability to read the contents of the file.
- **write** - The Write permissions refer to a user's capability to write or modify a file or directory.
- **execute** - The Execute permission affects a user's capability to execute a file or view the contents of a directory.

Viewing the Permissions

You can view the permissions by checking the file or directory permissions in your favorite GUI File Manager (which I will not cover here) or by reviewing the output of the ***\"ls -l\"*** command while in the terminal and while working in the directory which contains the file or folder.

The permission in the command line is displayed as: ***_rwxrwxrwx 1 owner:group***

1. User rights/Permissions
    1. The first character that I marked with an underscore is the special permission flag that can vary.
    2. The following set of three characters (rwx) is for the owner permissions.
    3. The second set of three characters (rwx) is for the Group permissions.
    4. The third set of three characters (rwx) is for the All Users permissions.
2. Following that grouping since the integer/number displays the number of hardlinks to the file.
3. The last piece is the Owner and Group assignment formatted as Owner:Group.

Modifying the Permissions

When in the command line, the permissions are edited by using the command ***chmod***. You can assign the permissions explicitly or by using a binary reference as described below.

Explicitly Defining Permissions

To explicity define permissions you will need to reference the Permission Group and Permission Types.

The Permission Groups used are:

- **u** - Owner
- **g** - Group
- **o** or **a** - All Users

The potential Assignment Operators are + (plus) and - (minus); these are used to tell the system whether to add or remove the specific permissions.

The Permission Types that are used are:

- **r** - Read
- **w** - Write
- **x** - Execute

So for an example, lets say I have a file named file1 that currently has the permissions set to **_rw_rw_rw,** which means that the owner, group and all users have read and write permission. Now we want to remove the read and write permissions from the all users group.

To make this modification you would invoke the command: ***chmod a-rw file1***

To add the permissions above you would invoke the command: ***chmod a+rw file1***

As you can see, if you want to grant those permissions you would change the minus character to a plus to add those permissions.

Using Binary References to Set permissions

Now that you understand the permissions groups and types this one should feel natural. To set the permission using binary references you must first understand that the input is done by entering three integers/numbers.

A sample permission string would be **chmod 640 file1**, which means that the owner has read and write permissions, the group has read permissions, and all other user have no rights to the file.

The first number represents the Owner permission; the second represents the Group permissions; and the last number represents the permissions for all other users. The numbers are a binary representation of the rwx string.

- ***r** = 4*
- ***w** = 2*
- ***x** = 1*

You add the numbers to get the integer/number representing the permissions you wish to set. You will need to include the binary permissions for each of the three permission groups.

So to set a file to permissions on file1 to read _***rwxr***_____, you would enter ***chmod 740 file1***.

Owners and Groups

I have made several references to Owners and Groups above, but have not yet told you how to assign or change the Owner and Group assigned to a file or directory.

You use the chown command to change owner and group assignments, the syntax is simple ***chown owner:group filename****,* so to change the owner of file1 to user1 and the group to family you would enter ***chown user1:family file1***.

Advanced Permissions

The special permissions flag can be marked with any of the following:

- _ - no special permissions
- ***d*** - directory
- ***l*** - The file or directory is a symbolic link
- ***s*** - This indicated the setuid/setgid permissions. This is not set displayed in the special permission part of the permissions display, but is represented as a **s** in the read portion of the owner or group permissions.
- ***t*** - This indicates the sticky bit permissions. This is not set displayed in the special permission part of the permissions display, but is represented as a **t** in the executable portion of the all users permissions

Setuid/Setgid Special Permissions

The setuid/setguid permissions are used to tell the system to run an executable as the owner with the owner\'s permissions.

Be careful using setuid/setgid bits in permissions. If you incorrectly assign permissions to a file owned by root with the setuid/setgid bit set, then you can open your system to intrusion.

You can only assign the setuid/setgid bit by explicitly defining permissions. The character for the setuid/setguid bit is **s**.

So do set the setuid/setguid bit on file2.sh you would issue the command ***chmod g+s file2.sh***.

Sticky Bit Special Permissions

The sticky bit can be very useful in shared environment because when it has been assigned to the permissions on a directory it sets it so only file owner can rename or delete the said file.

You can only assign the sticky bit by explicitly defining permissions. The character for the sticky bit is **t**.

To set the sticky bit on a directory named dir1 you would issue the command ***chmod +t dir1***.

When Permissions Are Important

To some users of Mac- or Windows-based computers you don't think about permissions, but those environments don't focus so aggressively on user based rights on files unless you are in a corporate environment. But now you are running a Linux-based system and permission based security is simplified and can be easily used to restrict access as you please.

So I will show you some documents and folders that you want to focus on and show you how the optimal permissions should be set.

- ***home directories*** - The users\' home directories are important because you do not want other users to be able to view and modify the files in another user\'s documents of desktop. To remedy this you will want the directory to have the **drwx_____ (700)** permissions, so lets say we want to enforce the correct permissions on the user user1\'s home directory that can be done by issuing the command **chmod 700 /home/user1**.
- ***bootloader configuration files*** - If you decide to implement password to boot specific operating systems then you will want to remove read and write permissions from the configuration file from all users but root. To do you can change the permissions of the file to 700.
- ***system and daemon configuration files*** - It is very important to restrict rights to system and daemon configuration files to restrict users from editing the contents, it may not be advisable to restrict read permissions, but restricting write permissions is a must. In these cases it may be best to modify the rights to 644.
- ***firewall scripts*** - It may not always be necessary to block all users from reading the firewall file, but it is advisable to restrict the users from writing to the file. In this case the firewall script is run by the root user automatically on boot, so all other users need no rights, so you can assign the 700 permissions.

Boot Security

Secure Boot is a technology where the system firmware checks that the system boot loader is signed with a cryptographic key authorized by a database contained in the firmware. With adequate signature verification in the next-stage boot loader(s), kernel, and, potentially, user space, it is possible to prevent the execution of unsigned code.

Secure Boot is a form of *Verified Booting*. Boot path validation is also part of other technologies such as *Trusted Boot*. Boot path validation is indepedent of secure storage of cryptographic keys and remote attestation.

1.1. UEFI Secure Boot

UEFI Secure Boot is the boot path validation component of the *UEFI* specification (*Unified Extensible Firmware Interface*)as of version 2.3. Roughly speaking, it specifies the following:

- a programming interface for cryptographically protected UEFI variables in non-volatile storage,
- how the trusted X.509 root certificates are stored in UEFI variables,
- validation of UEFI applications (boot loaders and drivers) using AuthentiCode signatures embedded in these applications, and
- procedures to revoke known-bad certificates and application hashes.

UEFI Secure Boot does not require specialized hardware, apart from non-volatile (flash) storage which can be switched from read-write mode to read-only mode during system boot. This storage has to be used to store the UEFI implementation itself and some of the protected UEFI variables (including the trusted root certificate store).
From a user point of view, a system which has enabled UEFI Secure Boot and which is confronted with a tampered boot path simply stops working until UEFI Secure Boot is disabled or a signed next-stage boot loader is available on boot media. (Figure 1.1, “Typical error message from UEFI Secure Boot” shows a typical error message.) Similarly, operating system installers without a cryptographically valid signature do not run and result in an error message. Users are not offered a way to override the boot loader decision to reject the signature, unlike the similar scenario with web server certificates. No certificate issuer information is provided to the user.

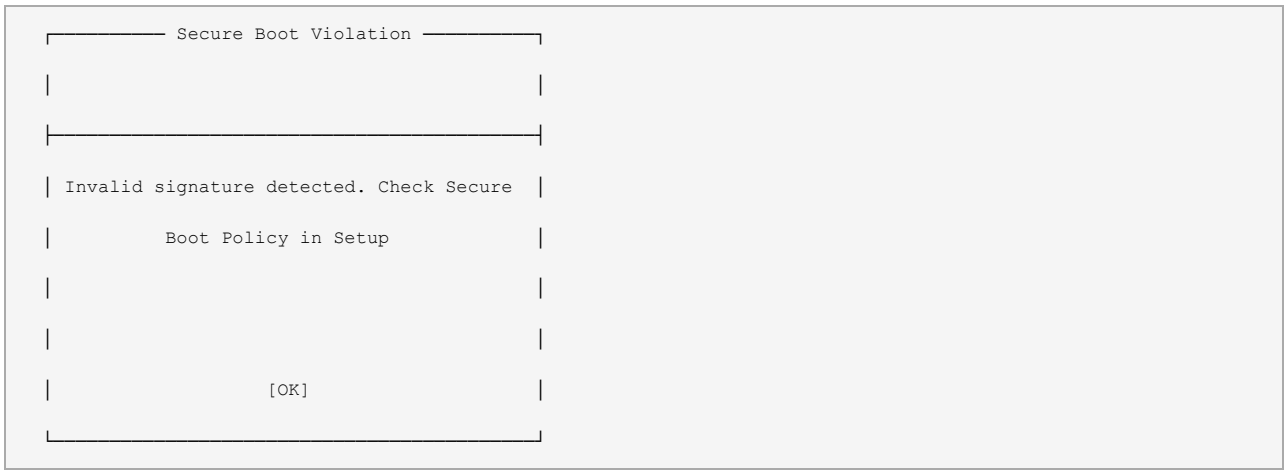


Figure 1.1. Typical error message from UEFI Secure Boot

UEFI Secure Boot does not prevent the installation or removal of second-stage boot loaders or require explicit user confirmation of such changes. Signatures are verified during booting, and not when the boot loader is installed or updated. Therefore, UEFI Secure Boot does not stop boot path manipulations. It only prevents the system from executing a modified boot path once such a modification has occurred, and simplifies their detection.

Security principles:

1)Network Based Security

Network security consists of the policies adopted to prevent and monitor unauthorized access, misuse, modification, or denial of acomputer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator.[citation needed] Users choose or are assigned an ID and password or other authenticating information that allows them access to information and programs within their authority. Network security covers a variety of computer networks, both public and private, that are used in everyday jobs; conducting transactions and communications among businesses, government agencies and individuals. Networks can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other types of institutions. It does as its title explains: It secures the network, as well as protecting and overseeing operations being done. The most common and simple way of protecting a network resource is by assigning it a unique name and a corresponding password.

Network security starts with authenticating, commonly with a username and a password. Since this requires just one detail authenticating the user name —i.e., the password— this is sometimes termed one-factor authentication. With two-factor authentication, something the user 'has' is also used (e.g., a security token or 'dongle', an ATM card, or a mobile phone); and with three-factor authentication, something the user 'is' also used (e.g., a fingerprint or retinal scan).

Once authenticated, a firewall enforces access policies such as what services are allowed to be accessed by the network users.[1]Though effective to prevent unauthorized access, this component may fail to check potentially harmful content such as computer worms or Trojans being transmitted over the network. Anti-virus software or an intrusion prevention system (IPS)[2] help detect and inhibit the action of such malware.

An anomaly-based intrusion detection system may also monitor the network like wireshark trafficand may be logged for audit purposes and for later high-level analysis.

Communication between two hosts using a network may be encrypted to maintain privacy.

1)Host Based Security

Here are some guidelines to follow in order to protect a machine/host while installing and using various operating systems and applications. These guidelines are not complete, but following these steps will help keep hosts secure as they are installed and used on the local network or the Internet. In general, the following steps should be taken for every host or device which is placed on the network, regardless of operating system. More detail on how to accomplish the various steps for each operating system are listed in later sections.

1. Install and configure a host based firewall
2. Choose good passwords for any accounts on the system, and change any default or well known accounts on the machine
3. Install and keep up with operating system patches and also hardware firmware patches
4. Configure and continue to monitor logs on the device
5. Disable services and accounts which are not being used, or are no longer necessary
6. Replace insecure services (such as telnet, rsh, or rlogin) with more secure alternatives such as ssh
7. Restrict access to services which cannot be disabled where possible
8. Make and test backups of the system in a consistent manner

Before Installing The Operating System

Before you begin any installation, be sure you have the following available:

1. Any operating system CDs or install media that you'll need, including license keys
2. A firewall to protect the host while you install everything (if the host will be connected to a network during the install) or all current patches and updates for the operating system on a CD (for a system which will not be connected to the network during installation)
3. A list of services the machine will provide, or a list of services which should be disabled after the machine is installed

A firewall is necessary to protect the host while you install the operating system and all necessary patches if you plan to have the host connected to a network during installation. Most operating systems are vulnerable to compromise when they are installed, and require many patches and updates before they can safely be allowed on the network. If a host is connected to the department network without direct protection,

even if the department has a firewall which protects the general network from outside problems, the host can still be compromised by another machine on the department network while it is being installed and configured.

If you choose to use a CD to install patches before you connect the machine to a network, make sure that the network cable is unplugged from the machine, and do not rely on software to prevent the machine from being accessible on the network.

By the end of 2003, the average survival time of an upatched, uncompromised Windows machine on the Internet is down to minutes, and under some circumstances seconds. Because it takes longer than this to install patches, an install must be done either behind a firewall or without a network connection.

The CS Department is blocking a variety of incoming ports in order to increase the security of hosts on the CS network. Despite these blocks, everyone is encouraged to have their own host based firewall.

Guidelines for installing various operating systems.

Linux

There are a number of distributions of Linux, and many use at least slightly different methods of package management, updates, and initialization scripts. The two major current distributions are RedHat and SuSE, although RedHat has split from a publicly available distribution, to Fedora (publicly available) and RedHat Enterprise series (commercial, restricted license). These will probably receive their own separate sections at some later point in time.

RedHat Linux

In general, RedHat includes a variety of scripts which can be used to start and stop various services, accessed through a user interface (such as Gnome or KDE, the two major desktop systems under Linux). In addition, the basic service startup scripts are accessed either via inetd, xinetd, or the standard UNIX init mechanism.

Upon installation, the install process will request that you choose a particular level for the firewall, which can generally be none (not recommended), medium, or high. Under each level, the firewall can and should be adjusted to allow incoming connections to system services which should be available to the Internet on the server. The firewall parameters can be further adjusted once the machine is installed, by altering the configuration files or using the graphical interface to adjust incoming services.

Workstations in general should use either the medium or the high settings, and should not allow any incoming services (again these settings can be adjusted later). The software used to control the firewall seems to change with almost every major new kernel version, so you should consult the manual pages for the current firewall system in use, or use a graphical interface (if it exists) to alter the firewall settings.

Both RedHat and SuSE currently use the /etc/rc.d/rcX.d directories to hold the init scripts, and general system configuration information can usually be found in the /etc/sysconfig directory.

The default root (administrator) environment under RedHat uses the Gnome desktop system, and there are a number of helper scripts which are used to enable and disable the various services and daemons. After you install the operating system, use the following guidelines to secure your RedHat system. While it's not as critical to do the following steps behind a firewall at this point when installing Linux, it is still a good idea to consider this before you begin.

1. Configure the RedHat up2date program to download updates.
2. Make sure you've installed a good password for any accounts which will have root capabilities (either directly or through sudo).
3. Verify that the RedHat firewall set up during the install is working. Check the current firewall configuration by selecting System Settings->Security Level and verify the security level is set to either “Medium” or “High”. If you need to customize the settings for particular services, change “Use default firewall rules” to “Customize” and adjust the ports in the lower section to allow inbound connections to certain services.
4. Disable system processes which are not in use and prevent them from restarting. For Gnome the procedure is something like : 1. Select Server Settings->Services 2. Select the service you wish to affect by unchecking its checkbox 3. Press the “Stop” button to halt the process that is currently running (some processes may not be stoppable directly, if they run out of the xinetd process you should disable the xinetd process as well) 4. After doing this with all processes you wish to disable, choose Save from the File menu.

What is SSL and what are Certificates?

The Secure Socket Layer protocol was created by Netscape to ensure secure transactions between web servers and browsers. The protocol uses a third party, a Certificate Authority (CA), to identify one end or both end of the transactions. This is in short how it works.

1. A browser requests a secure page (usually https://).

2. The web server sends its public key with its certificate.
3. The browser checks that the certificate was issued by a trusted party (usually a trusted root CA), that the certificate is still valid and that the certificate is related to the site contacted.
4. The browser then uses the public key, to encrypt a random symmetric encryption key and sends it to the server with the encrypted URL required as well as other encrypted http data.
5. The web server decrypts the symmetric encryption key using its private key and uses the symmetric key to decrypt the URL and http data.
6. The web server sends back the requested html document and http data encrypted with the symmetric key.
7. The browser decrypts the http data and html document using the symmetric key and displays the information.

Several concepts have to be understood here.

1.2.1. Private Key/Public Key:

The encryption using a private key/public key pair ensures that the data can be encrypted by one key but can only be decrypted by the other key pair. This is sometime hard to understand, but believe me it works. The keys are similar in nature and can be used alternatively: what one key encrypts, the other key pair can decrypt. The key pair is based on prime numbers and their length in terms of bits ensures the difficulty of being able to decrypt the message without the key pairs. The trick in a key pair is to keep one key secret (the private key) and to distribute the other key (the public key) to everybody. Anybody can send you an encrypted message, that only you will be able to decrypt. You are the only one to have the other key pair, right? In the opposite , you can certify that a message is only coming from you, because you have encrypted it with you private key, and only the associated public key will decrypt it correctly. Beware, in this case the message is not secured you have only signed it. Everybody has the public key, remember!

One of the problem left is to know the public key of your correspondent. Usually you will ask him to send you a non confidential signed message that will contains his publick key as well as a certificate.

```
Message-->[Public Key]-->Encrypted Message-->[Private Key]-->Message
```

1.2.2. The Certificate:

How do you know that you are dealing with the right person or rather the right web site. Well, someone has taken great length (if they are serious) to ensure that the web site owners are who they claim to be. This someone, you have to implicitly trust: you have his/her certificate loaded in your browser (a root Certificate). A

certificate, contains information about the owner of the certificate, like e-mail address, owner's name, certificate usage, duration of validity, resource location or Distinguished Name (DN) which includes the Common Name (CN) (web site address or e-mail address depending of the usage) and the certificate ID of the person who certifies (signs) this information. It contains also the public key and finally a hash to ensure that the certificate has not been tampered with. As you made the choice to trust the person who signs this certificate, therefore you also trust this certificate. This is a certificate trust tree or certificate path. Usually your browser or application has already loaded the root certificate of well known Certification Authorities (CA) or root CA Certificates. The CA maintains a list of all signed certificates as well as a list of revoked certificates. A certificate is insecure until it is signed, as only a signed certificate cannot be modified. You can sign a certificate using itself, it is called a self signed certificate.

The Symmetric key:

Well, Private Key/Public Key encryption algorithms are great, but they are not usually practical. It is asymmetric because you need the other key pair to decrypt. You can't use the same key to encrypt and decrypt. An algorithm using the same key to decrypt and encrypt is deemed to have a symmetric key. A symmetric algorithm is much faster in doing its job than an asymmetric algorithm. But a symmetric key is potentially highly insecure. If the enemy gets hold of the key then you have no more secret information. You must therefore transmit the key to the other party without the enemy getting its hands on it. As you know, nothing is secure on the Internet. The solution is to encapsulate the symmetric key inside a message encrypted with an asymmetric algorithm. You have never transmitted your private key to anybody, then the message encrypted with the public key is secure (relatively secure, nothing is certain except death and taxes). The symmetric key is also chosen randomly, so that if the symmetric secret key is discovered then the next transaction will be totally different.

```
Symetric Key-->[Public Key]-->Encrypted Symetric Key-->[Private Key]--
>Symetric Key
```

1.2.4. Encryption algorithm:

There are several encryption algorithms available, using symmetric or asymmetric methods, with keys of various lengths. Usually, algorithms cannot be patented, if Henri Poincare had patented his algorithms, then he would have been able to sue Albert Einstein... So algorithms cannot be patented except mainly in USA. OpenSSL is developed in a country where algorithms cannot be patented and where encryption technology is not reserved to state agencies like military and secret services. During the negotiation between browser and web server, the applications will indicate to each other a list of algorithms that can be understood ranked by order of preference. The common preferred algorithm is then chosen.

OpenSSL can be compiled with or without certain algorithms, so that it can be used in many countries where restrictions apply.

1.2.5. The Hash:

A hash is a number given by a hash function from a message. This is a one way function, it means that it is impossible to get the original message knowing the hash. However the hash will drastically change even for the slightest modification in the message. It is therefore extremely difficult to modify a message while keeping its original hash. It is also called a message digest. Hash functions are used in password mechanisms, in certifying that applications are original (MD5 sum), and in general in ensuring that any message has not been tampered with. It seems that the Internet Enginering Task Force (IETF) prefers SHA1 over MD5 for a number of technical reasons (Cf RFC2459 7.1.2 and 7.1.3).

1.2.6. Signing:

Signing a message, means authentifying that you have yourself assured the authenticity of the message (most of the time it means you are the author, but not neccesarily). The message can be a text message, or someone else's certificate. To sign a message, you create its hash, and then encrypt the hash with your private key, you then add the encrypted hash and your signed certificate with the message. The recipient will recreate the message hash, decrypts the encrypted hash using your well known public key stored in your signed certificate, check that both hash are equals and finally check the certificate.

The other advantage of signing your messages is that you transmit your public key and certificate automatically to all your recipients.

There are usually 2 ways to sign, encapsulating the text message inside the signature (with delimiters), or encoding the message altogether with the signature. This later form is a very simple encryption form as any software can decrypt it if it can read the embedded public key. The advantage of the first form is that the message is human readable allowing any non complaint client to pass the message as is for the user to read, while the second form does not even allow to read part of the message if it has been tampered with.

1.2.7. PassPhrase:

“A passprase is like a password except it is longer”. In the early days passwords on Unix system were limited to 8 characters, so the term passphrase for longer passwords. Longer is the password harder it is to guess. Nowadays Unix systems use MD5 hashes which have no limitation in length of the password.

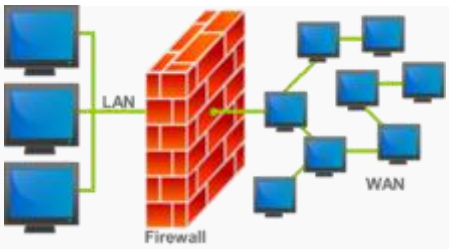
1.2.8. Public Key Infrastructure

The Public Key Infrastructure (PKI) is the software management system and database system that allows to sign certifcate, keep a list of revoked certificates, distribute public key,... You can usually access it via a website and/or ldap server. There will be also some people checking that you are who you are... For securing individual applications, you can use any well known commercial PKI as their root CA certificate is most likely to be inside your browser/application. The problem is for securing e-mail, either you get a generic type certificate for your e-mail or you must pay about USD100 a year per certificate/e-mail address. There is also no way to find someone's public key if you have never received a prior e-mail with his certificate (including his public key).

Firewall (computing)

In computing, a **firewall** is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules.[1] A firewall typically establishes a barrier between a trusted, secure internal network and another outside network, such as the Internet, that is assumed to not be secure or trusted.[2] Firewalls are often categorized as either*network firewalls* or *host-based firewalls*. Network firewalls are a software appliance running on general purpose hardware or hardware-based firewall computer appliances that filter traffic between two or more networks. Host-based firewalls provide a layer of software on one host that controls network traffic in and out of that single machine.[3][4] Firewall appliances may also offer other functionality to the internal network they protect such as acting as a DHCP[5][6] or VPN[7][8][9][10] server for that network.[11]

Types[edit]



An illustration of where a firewall would be located in a network

There are different types of firewalls depending on where the communication is taking place, where the communication is intercepted and the state that is being traced.[27]

Network layer or packet filters[edit]

Network layer firewalls, also called packet filters, operate at a relatively low level of the TCP/IP protocol stack, not allowing packets to pass through the firewall unless they match the established rule set. The firewall administrator may define the rules; or default rules may apply. The term "packet filter" originated in the context of BSD operating systems.

Network layer firewalls generally fall into two sub-categories, stateful and stateless. Stateful firewalls maintain context about active sessions, and use that "state information" to speed packet processing. Any

existing network connection can be described by several properties, including source and destination IP address, UDP or TCP ports, and the current stage of the connection's lifetime (including session initiation, handshaking, datatransfer, or completion connection). If a packet does not match an existing connection, it will be evaluated according to the ruleset for new connections. If a packet matches an existing connection based on comparison with the firewall's state table, it will be allowed to pass without further processing.

Stateless firewalls require less memory, and can be faster for simple filters that require less time to filter than to look up a session. They may also be necessary for filtering stateless network protocols that have no concept of a session. However, they cannot make more complex decisions based on what stage communications between hosts have reached.

Newer firewalls can filter traffic based on many packet attributes like source IP address, source port, destination IP address or port, destination service like WWW or FTP. They can filter based on protocols, TTL values, netblock of originator, of the source, and many other attributes.

Commonly used packet filters on various versions of Unix are *IPFilter* (various), *ipfw* (FreeBSD/Mac OS X), *NPF* (NetBSD), *PF* (OpenBSD, and some other BSDs),*iptables*/*ipchains* (Linux).

Application-layer[edit]

Main article: Application layer firewall

Application-layer firewalls work on the application level of the TCP/IP stack (i.e., all browser traffic, or all telnet or ftp traffic), and may intercept all packets traveling to or from an application. They block other packets (usually dropping them without acknowledgment to the sender).

On inspecting all packets for improper content, firewalls can restrict or prevent outright the spread of networked computer worms and trojans. The additional inspection criteria can add extra latency to the forwarding of packets to their destination.

Application firewalls function by determining whether a process should accept any given connection. Application firewalls accomplish their function by hooking into socket calls to filter the connections between the application layer and the lower layers of the OSI model. Application firewalls that hook into socket calls are also referred to as socket filters. Application firewalls work much like a packet filter but application filters apply filtering rules (allow/block) on a per process basis instead of filtering connections on a per port basis. Generally, prompts are used to define rules for processes that have not yet received a connection. It is rare to find application firewalls not combined or used in conjunction with a packet filter.[28]

Also, application firewalls further filter connections by examining the process ID of data packets against a ruleset for the local process involved in the data transmission. The extent of the filtering that occurs is defined by the provided ruleset. Given the variety of software that exists, application firewalls only have more complex rulesets for the standard services, such as sharing services. These per process rulesets have limited efficacy in filtering every possible association that may occur with other processes. Also, these per process rulesets cannot defend against modification of the process via exploitation, such as memory corruption exploits. Because of these limitations, application firewalls are beginning to be supplanted by a new generation of application firewalls that rely on mandatory access control (MAC), also referred to as sandboxing, to protect vulnerable services.[29]

Proxies[edit]

Main article: Proxy server

A proxy server (running either on dedicated hardware or as software on a general-purpose machine) may act as a firewall by responding to input packets (connection requests, for example) in the manner of an application, while blocking other packets. A proxy server is a gateway from one network to another for a specific network application, in the sense that it functions as a proxy on behalf of the network user.[2]

Proxies make tampering with an internal system from the external network more difficult and misuse of one internal system would not necessarily cause a security breach exploitable from outside the firewall (as long as the application proxy remains intact and properly configured). Conversely, intruders may hijack a publicly reachable system and use it as a proxy for their own purposes; the proxy then masquerades as that system to other internal machines. While use of internal address spaces enhances security,crackers may still employ methods such as IP spoofing to attempt to pass packets to a target network.

Network address translation[edit]

Main article: Network address translation

Firewalls often have network address translation (NAT) functionality, and the hosts protected behind a firewall commonly have addresses in the "private address range", as defined in RFC 1918. Firewalls often have such functionality to hide the true address of protected hosts. Originally, the NAT function was developed to address the limited number of IPv4 routable addresses that could be used or assigned to companies or individuals as well as reduce both the amount and therefore cost of obtaining enough public addresses for every computer in an organization. Hiding the addresses of protected devices has become an increasingly important defense against network reconnaissance.[30]