CHAPTER 8

Limiting Access

Halting the hacker before he gains access to your system requires a strong perimeter defense. At every spot where a hacker can gain access, you must put up defenses that require more than modest authentication. These spots should also have a fall-back detection scheme for all access and a notification scheme for those that are out of the ordinary.

A policy of least privileges should be enforced. No user needs more privileges than the privileges needed to do his job. A system need not provide services in excess of what are necessary for the proper function of that system. The key to managing this environment successfully is to keep abreast of what the users need to do their jobs and respond rapidly to supply these needs. This is a very difficult task given the rapidly changing environment in which most companies find themselves.

This is why most nongovernment organizations implement more relaxed policies.

Physical System Access

Physical security is fundamental to the security of an information system. Physical access to a system is the most compromising of any access issues. With physical access, one can physically damage the system, remove data storage devices, and, in most cases, gain access to the system's operations. Physical access needs to be limited to those who have a need to physically access the systems and it must produce a record of who had physical access and when.

Numerous sites have been plagued with stolen equipment and, in many cases, it has been memory modules, CPUs, and other internal components which have been stolen. This illustrates a need for locks on the computer cabinets as well as auditable locks on the rooms which house the computer systems. These locks must be able to identify the individual who entered and exited the room. This is usually accomplished with individually issued access cards. Integration of physical security into information security can simplify the management, monitoring, and correlation of security events.

This decentralization of systems means that more than just the computer operations staff must be made aware of physical security. The users must understand the importance of locking up their floppies and printouts and logging off or locking up their computers. A proprietary report requires the same security whether it is printed or on a diskette or in your computer.

Mobile computing users who use portable computers must be aware that the theft of their computer compromises all the information on that computer. Many corporate spies have found it easier to steal a portable computer than to break into a company's computer to get the desired information.

> An employee garnered a philanthropic reputation for treating his coworkers to pizza in the cafeteria on Friday afternoons. But while his colleagues munched on pepperoni, the employee's accomplice — the pizza man — stuffed his delivery bag with laptop computers and left the building without being challenged. [53]

Telecommuters, those employees who work from home, also add additional security issues, and they need the same level of security at home as they would have in the office.

Wireless communication adds a whole new area for hackers to exploit by eavesdropping without physical access. Cellular modems and wireless local area networks (LANs) have opened the doors to your data communication without a hacker having to physically attach to your network.

If your physical security procedures have not recently been reviewed, they should be. It is extremely important to review security procedures regularly to incorporate new equipment and technologies.

Network Equipment

The computer system extends beyond the physical box which contains it. The network to which it is connected also requires physical security for the network equipment and the junction points. Often network equipment is located in wiring closets which are shared with other functions, such as telephone equipment. The security of these areas are equally important to the security of the information system. Access to these closets are often given to repair personnel of other companies, such as telephone repairmen. This is why audited and monitored access is required. Unauthorized access to these areas enables a hacker to add unauthorized connections for eavesdropping or unauthorized access or to disable connections, causing disruptions in service.

[53] Mello, Jr., John P., "Stop, Thief!," *CFO Magazine*, 1 October 1997.

Removable Media

Today, with tape backup technology allowing gigabytes of data to be stored on a tape that can fit into a shirt pocket, an entire data center's information can easily slip past normal physical security procedures. Much more attention than ever before must be paid to the sites that have removable media. With the decentralization of systems, removable media are everywhere.

Physical access precautions should be extended to anywhere there are removable media. Access to removable media devices allows a hacker to remove information from the system and the site without going through the network security which might otherwise detect it. Physical secure checkpoints are rarely adequate to detect the removal of information on removable media. The data density and physical size of tapes today make it easy to slip huge amounts of data past security checkpoints.

Once information is removed from the information system, the protection afforded by that information system, such as user authorizations and file permissions, does not apply. The information on removable media can be installed onto any system without regard to the security level or controls of that system.

Removable media are always a security issue. They create a porthole through which information can flow out of and into the system. Restricting the programs that can access these devices and the people who can run these programs will help limit this flow of information. Restricting the number of devices that have removable media and the number of people who have physical access to those devices will also help reduce the risk. Physical security will help limit this threat. So can appropriate labeling procedures.

It is wise to produce custom "company" labels that are specific to the classification of the data they will contain. When output devices are limited to a specific classification level, this can make for easy and rapid identification if the data are being handled correctly. In conjunction with a widespread employee security awareness program, this very simple concept can make a big difference in spotting inappropriate handling of information and having it reported.

System Backups

Backups can be both a blessing and a curse to the hacker as well as to the system manager. For a hacker, if gaining access to backups is easy, then accessing information from them may be easier than getting the same information from the system's disks. However, if the hacker's activity is logged and backed up, that may be just the evidence it takes to convict him. For you as a system manager, backups are your last safety net. You can never lose more data than that which has been created since your last good backup. When backups are stored off-site, you can recover from a physical disaster. However, if your backup and recovery policies are not sufficient, a hacker may be able to access your system's information from the backup or restore hacker code onto your system.

There is no bigger risk to your information systems than your system backups. Your entire system is on those tapes. If they are compromised, all the information contained on your system

is compromised. Proper handling and storage of backups are critical to ensure the confidentiality and integrity of the information they contain. Nowhere are procedures more important than in the handling of removable media. Backups must be kept in a secure area. Anyone who has physical access to your backups can read them on another computer. If your backups are stored off-site, the transportation to and from the off-site storage must also be a secure process.

Procedures to request the mounting of backup media must be secure. This means a separate authentication of the requester. You need to understand the backup policy and procedures for your system, keeping in mind how they might be used by a hacker who plans to use your system to attack other systems or to plunder the information on your system.

Restricting Users

Reducing the number of user accounts on a system reduces the number of entry points into the system which can be exploited. Every authorized user must maintain the secrecy of his password and the administrator must provide a secure user environment and services. User accounts are often used by intruders as a place from which to work. Minimizing the capabilities of user accounts will minimize the intruder's ability to misuse the accounts.

Privileged Users

Privileged users are those accounts which are granted privileges above the minimum set of privileges for all accounts. These additional privileges give the user extra capabilities and are targets of hackers so that they can utilize these extra capabilities.

The root user, UID 0, on UNIX systems is an all-powerful user who does not have to obey the security restrictions which apply to all other users. For this reason, the root user's access should be limited to the system.

Securetty is a UNIX feature which defines secure terminals which the root user can use. A secure terminal is a configuration that limits the superuser's access to the system to a list of terminals that are considered secure. This limits only the login process. A user may still log in as a regular user and change to the superuser with the su command. Other access methods, such as X windows, do not adhere to the secure terminal settings.

It is advisable that root not be allowed to log in from any unsecured terminal. Only the system console, `/dev/console`, should be considered secure. Instead, require that users log in as themselves and switch user to root, thereby giving you a log of who is root. You can restrict root's access by using the secure terminal facility. This facility is implemented in a number of different ways, depending on the version of the UNIX operating system.

If the system is a System V derivative, such as Linux or HP-UX, it will have a file, `/etc/securetty` or `/etc/default/login` on Solaris, which will contain a list of the terminals that are considered secure. Only directly connected terminals whose connection does not leave the secured area should be included as a secure terminal. Often the console should be the only secure terminal.

If the implementation is a Berkeley Software Distribution (BSD) derivative, then you will need to add the secure parameter to the console line in the `/etc/ttys` or `/etc/ttytab` file. This line will look something like the following:

```
console "/etc/getty std.9600" vt100 on local secure
```

Time-based Access Control

With time-based access controls, the system administrator may specify times-of-day and days-of-week that are allowed for login for each user. When a user attempts to log in outside the allowed access time, the event is logged (if auditing is enabled for login failures and successes) and the login is terminated.

Most UNIX systems will implement time-based access controls through PAM features.

On HP-UX, enabling trusted systems enables this feature. A superuser can log in outside the allowed access time, but the event is logged. The permitted range of access times is stored in the protected password database for users and may be set with SAM. Users that are logged in when a range ends are not logged out.

Device-based Access Control

In the early days of interactive computing, the only connection to the computer was over serial lines. These serial lines allowed users to access the computer with terminals that were directly connected to the serial line or through a modem. Adding dialers to the modems allowed computers to dial out as well as receive incoming calls. Soon computers were calling other computers and serial networking was born.

Today most multiuser systems still use serial line access. The system console is almost always directly connected to the system by a serial line. If the system is a data entry system, it is likely that most of the users are using directly connected terminals. Also, many systems have modems attached to them. These are almost always attached via serial lines.

The `/etc/securetty` configuration file can be used to limit to specific terminals on which the root user can log in. The login program on Red Hat and Mandrake Linux also uses the `/etc/usrtty` configuration file to define from which terminals and from which remote systems each user can log in.

Dial-up Access

Trying to guess logins and passwords is the most dangerous and unproductive way for a hacker to access a system. This type of bell ringing, the repeated calling of a modem and attempting to login, will undoubtedly be noticed by any system manager. So unless the hacker has some insight into logins and passwords, it is unlikely that he will get anywhere. The attempts to log in will be logged, whether they are successful or not, by the accounting system.

Correctly configured, the system will hang up the modem after three failed login attempts and will also terminate a session that is disconnected.

For the login program to hang up the modem, the modem must be appropriately connected to the system, with a cable that supports full modem control, and attached to a port on the computer that also supports full modem control. The entry for the modem connection in `/etc/gettydefs` must also be correctly configured. The gettydefs entry must have a HUPCL (Hang-Up and Clear) entry in both sections of the entry as shown:

```
2400 # B2400 HUPCL IGNPAR ICRLN IXON OPOST ONLCR CS8
    CREAD ISIG ICANON ECHO ECHOK ISTRIP IXANY TAB3
    # B2400 HUPCL SANE CS8 ISTRIP IXANY TAB3
    # login:  #2400
```

Even with all these precautions, you must understand that phone lines are not secure: Some hackers are also able to hack the phone network, thereby being able to eavesdrop on communications, reroute calls, or steal a phone connection right out from under you, so the hacker is now connected to the session to which you were connected.

The ability to trace access through a dial-up line is now increased, since Caller ID® is available in most areas. This means it is no longer necessary to get a court order or file a harassing caller complaint to get the phone number of the calling individual.

Today Caller IDTM devices are available that will record the time and phone number of all the calls received. There are also Caller ID modems that present the computer with the phone number prior to connecting. These can be used to further authenticate users for a dial-back system, or to notify someone when there is an unauthorized access in progress.

One way to enhance the authentication is through dial-up security. This feature, which is implemented on some versions of UNIX systems, asks the user for two passwords. Even though this is referred to as dial-up security, it can be applied to any terminal or modem port on a port-by-port basis. The first password requested is the user's password; the second is a password based on the user's default shell. This requires the configuring of two files: `/etc/dialups` and `/etc/d_passwd`. The dialups file contains the list of ttys, terminal ports, on which the second password will be required.

```
/dev/tty0p0
/dev/tty0p1
```

The d_password file contains the default shell and the encrypted password for that shell. If a user's shell is not listed in this file, the dial-up password is not tested.

```
/bin/sh:dpscen80aKWa2:
/bin/ksh:dpJm/BwWmbsJg:
```

One of the difficulties with maintaining the d_passwd file is the process of encryption of the passwords. Here is a program that will encrypt a password given on the command line and write the encryption to standard out. This can be used to create the encrypted password in the d_passwd file.

```
main(argc,argv)
int argc;
char **argv;
{
  char *salt="dp"; /* use your favorite salt */
  printf ("%s",crypt(argv[1], salt);
}
```

Direct-connectTerminals

Compared to dial-up lines, direct-connect terminals have two disadvantages in keeping out a hacker. The first is a minor disadvantage — not having to redial after three failed login attempts. The second is a very real disadvantage — the ability of a hacker to get access to an unattended session when terminals are logged on and left alone. In a matter of seconds, a hacker can utilize an unattended session to gain access to a system or to gain privileges.

All inactive sessions should be logged off. This is generally more difficult than simply setting the shell time-out variable to a time-out value. This is because some programs' activity will not be measured and some applications will appear active to the shell even if there is no user interaction. In these cases, the terminal is allowed to lock while it is being used or to not lock when it is not being used. Some terminal locking software that has the ability to spawn a "screen saver" program may open back doors by using this feature if the spawned programs are not well-constructed.

On an HP-UX Trusted System, the system administrator can specify a list of users allowed for access each dedicated port on a MUX or DTC. When the list is null for a device, all users are allowed access. The device access information is stored in the device assignment database, `/tcb/files/devassign`, which contains an entry for each terminal device on the Trusted System. A field in the entry lists the users allowed on the device.

Over the Network

Attacking a system over the network gives the hacker a significant advantage over a serial line. On the serial line, the hacker has only the login program, and possibly UUCP, to attack. However, over the network, he has a plethora of programs offering services through different sockets on the network. Most of these services use simple text-based protocols that can be attacked, even if the hacker has only terminal access.

Dial-up access directly into a network is available only via a terminal server or a dial-up network server utilizing a serial line protocol, such as SLIP or PPP.

Terminal/Modem Servers

Network terminal or modem servers are devices that are directly attached to the network and allow for either direct-connected terminals or modems. The connection through these devices will generally give the user a prompt that allows him to connect to any device on the network. This connection will usually use either telnet or a proprietary protocol. In either case, the remote computer will see the user as a simple terminal connecting over the network. Many universities and businesses use terminal servers to consolidate the costs of modems and telephone lines into one location that can be utilized by everyone in the organization.

In many cases, a hacker will get a connection by simply connecting to the modem server. There may be no password required. He may then be able to connect to any computer that is on the same network or any system on which there is routing information. In either case, network terminal/modem servers are a very useful commodity to the hacker. Some of these network terminal servers will allow him to connect to the modem that is attached to the port and dial out using that modem. If that is the case, he has the ability to dial in and dial out, allowing him to put the long-distance call on your bill and to do connection laundering; that is, anyone who is tracing his activities to where he dialed out will come back to you, the owner of the terminal server, instead of directly to the hacker.

If you can require some level of authentication on your terminal server, do so. Giving free access to your network is asking for trouble. Restrict the systems to which the terminal server can connect. This will reduce your level of vulnerability. Utilize Caller ID on all modems. Institute callback security where possible. Where possible, do not allow dial-out from terminal servers.

Dial-up SLIP/PPP Servers

Today it is common to want to extend your network so you can facilitate users who work on the road or at home. This is usually done by having a dial-up SLIP or PPP server. This server gives TCP/IP connectivity to the system that dials into it.

This will allow the hacker to be a peer on the network. A hacker's system can utilize all the network tools at his disposal to probe systems. Gaining access over the network is much easier than over a terminal line. However, gaining access to a dial-up SLIP or PPP connection will generally be more difficult than a simple text connection. Text connections are often guarded by only a login ID and password. The dial-up SLIP will also require IP address information. Organizations should put stronger security on SLIP and PPP connections. These should include a hardware-based password system and some type of smart card, so access is not possible without physically having the smart card. This is termed two-factor authentication, because it is based on something you know, a password or PIN number, and something you have, a smart card or

authentication token. It is also a very good idea to have Caller ID enabled on all dial-up connections.

Host-based Firewalls

A host-based firewall is a software product which evaluates each network packet that the system receives and determines if it should accept it, based on a variety of packet features including the source address and packet type. They have the ability to evaluate all types of packets, including UDP and ICMP.

When an IP packet is received, the software goes down a list of rules until it finds a rule matching the packet and then handles the packet in the manner that the rule specifies.

IPChain is a stateless firewall. This means the determination of accepting the packet is based solely on its source and destination addresses, port number, and protocol.

System administrators who already employ ipchain-based firewalls should begin to migrate their scripts to iptables before the release of 7.2. Red Hat 7.1 comes preinstalled with a 2.4.x kernel that has netfilter and iptables compiled in.

The following shell script configures ipchain to deny all access except SSH.

```
#!/bin/sh
PATH=/usr/sbin:/sbin:/bin:/usr/sbin
LOCAL_INTERFACE="192.168.1.1/32" #  IP address
LOCAL_NETWORK="192.168.1.0/24"   #  IP address/mask here
SSH_PERMITTED="192.168.1.2/32 192.168.2.3/32" # who allowed to ssh
# deny everything
ipchains -P input DENY
ipchains -P output DENY
ipchains -P forward DENY
ipchains -F
#permit ssh
for ipaddr in $SSH_PERMITTED;
do
ipchains -A input -p tcp -s $ipaddr -i $LOCAL_INTERFACE -j ACCEPT
done
# permit outgoing tcp
ipchains -A output -p tcp -i $LOCAL_INTERFACE -j ACCEPT
ipchains -A input -p tcp ! -y -i $LOCAL_INTERFACE -j ACCEPT
# all the other connection attempts
ipchains -A input -p tcp -i $LOCAL_INTERFACE -l -j DENY
ipchains -A input -p udp -i $LOCAL_INTERFACE -l -j DENY
ipchains -A input -p icmp -i $LOCAL_INTERFACE -l -j DENY
```

IPTables is part of the netfilter project and the replacement for ipchains in the Linux 2.4 kernel. Iptables has many more features than ipchains, including the ability to do stateful packet inspection, and a clean separation of packet filtering and network address translation.

```
#!/bin/sh
iptables -F
# permit outgoing connections
iptables -P OUTPUT ACCEPT
# deny inbound connections
iptables -P INPUT DROP
# allow packets on loopback interface
iptables -A INPUT lo -h ACCEPT
```

IPFilter is a stateful firewall package. It maintains session information so that it can associate each individual packet to the session to which it belongs. This allows for better selection of allowing or denying packets, since the additional information about the session is available. This session information can also be applied to sessionless connections, such as UDP and ICMP. IPFilter will associate these packets into a virtual session to which they belong and provide the additional security based on this session information.

IPFilter is was built on BSD based UNIX systems and is available on Solaris and Irix as well as HP-UX.

IPFilter/9000 (B9901AA) is a port to HP-UX of the popular BSD IPFilter program, which is a public-domain stateful inspection host-based firewall system. It provides for the filtering of selected IP traffic into or out of the system. The traffic can be selected by source address, destination address, protocol port number, packet features, or any combination of these. It is provided for use as a system firewall on hosts running HP-UX 11i.

A system firewall is a packet filtering mechanism that is built into the TCP/IP stack of a host and provides filtering functionality specifically configured for the protection of that particular host. This program uses a sophisticated stateful-inspection packet filtering technology to filter traffic that enters or exits an individual HP-UX host.

Multi-homed HP-UX systems can be configured to discard incoming packets that are received through one network interface but whose destination address is that of a different interface of the same host, as well as to block the sending of outgoing packets whose source address is not that of the interface through which they are being sent. This packet filtering feature characterizes the Strong End-System (ES) functionality described in RFC 1122 of the IETF.

It can also function as a limited application proxy, but it is not recommended or supported as a general-purpose application proxy.

Designed to be used as a firewall, it is quite capable of being used to protect a host from network attacks. By default, the product will allow all packets to pass both in and out. However, by adding the appropriate filters to `/etc/opt/ipf/ipf.conf`, all packets can be blocked.

It is supported on HP-UX 11, with appropriate patches, in both 32- and 64-bit mode. It is released as a no-charge software product on the application CD.

The following configuration file denies all except SSH:

```
# By default block all packets
block in all
block in proto tcp all flags s/sa
block in proto udp all
block in proto icmp all
# Allow packets on loopback interface
pass in quick on lo0 all
pass out quick on lo0 all
# Block all packets with IP options
block in log quick all with opt lsrr
block in log quick all with opt ssrr
block in log quick all with ipopts
# Block all packets with a length which is too short to be real
block in log quick proto tcp all with short
# pass secure shell
pass in on le0 proto tcp from 192.168.1.2/32 port = 22 keep state
# allow all outbound connections, initiated by me.
pass out quick proto tcp from any to any flags S keep state keep frags
pass out quick proto udp from any to any keep state
pass out quick proto icmp from any to any keep state
```

Packet Filtering

Packet filtering is a method of restricting network access based on the network service being requested and the hosts requesting the service. On specific machines, this is accomplished by disabling the service, using a wrapper program to deny access to the service, or using the internet daemon's security to limit the hosts that can use the service on systems that support it. Usually, you will want to do packet filtering on a network level instead of a host level. This can be accomplished with filtered bridges or routers. If your site isn't filtering certain TCP/IP packets, it may not be as secure as you think it is.

System managers, security managers, and network managers need to understand packet filtering issues. Due to the flaws in several TCP/IP services and chronic system administration problems, a site must be able to restrict external access to these services. It is recommended that the following services be filtered:

- **DNS zone transfers** can be used by hackers to request all the information contained in the Domain Name Services database. Permit access to this service only from known secondary domain name servers. This will prevent intruders from gaining additional knowledge about the systems connected to your local network.
- **TFTP** allows unauthenticated access to a system and lets the hacker put files on and get files from the system. A system with TFTP enabled can be used as a depot for the transfer of information or stolen information.

- **SunRPC** supports all the ONC®-based services.
- **NFS (**Network File System) has long been used by hackers to gain information and access to systems through inappropriate configuration and software problems.
- **rexec** is used to execute a program remotely. It always requires a password and leaves a minimal amount of log information. It is used by hackers who have initially compromised a system so they can regain access without leaving tracks.
- **rlogin** (the remote login service) uses Berkeley Trusted Hosts configuration and security.
- **rsh** remotely executes a program using trusted systems configuration and security.

  Both rlogin and rsh are used by hackers with the use of personal .rhosts files to create an intricate web of connections between systems and users on those systems. This can make it extremely difficult to track the hacker back to his origin through dozens of different machines, where he has utilized different user IDs on each one.
- **lpd (**remote printer daemon) allows unauthenticated access to the system's print spooler resources.
- **uucpd** allows UUCP to run over the network. Running this services opens all the UUCP security issues over the network.
- **X Windows** windowing system has been utilized to allow eavesdropping and capturing the keystrokes of the user on the system.

There are a variety of network analysis tools that will determine which sockets a system has active. These include SATAN and strobe.

If the site does not need to provide other services to external users, those other services should be filtered.

Restricting Services

Restricting the services that allow access to a system reduces the methods by which a system can be attacked. Exploiting vulnerabilities in services is the most common method for gaining access to a system, so if a system has fewer services, then there is less code to contain vulnerabilities.

Anonymous Access

Anonymous access is any access to the service without authenticating an identifier. Historically, there has been a number of anonymous services on UNIX machines. However, as the machines have become more reliable and the need for security has increased, these anonymous information services have dropped away. Today, there are few anonymous services.

Anonymous FTP is a configuration of FTP which allows someone who does not have an account on the machine to FTP files to and from the machine anonymously. This is done by entering either "anonymous" or "ftp" as the user name and anything as a password; by convention, this is the requester's e-mail address. Even though it is called anonymous and there is no user authentication, it is not always anonymous. There are a number of FTP daemons that log file transfer and the location from where the request was made. Some FTP daemons will validate that the given password is a valid user on the system that is making the request.

These transfers are restricted to the home directory tree of the user named "ftp." Anonymous FTP needs a partial file system configured in its home directory since it is a "chroot"ed process. This partial file system includes a password file. Some automated administration scripts, when creating the password file for anonymous FTP, will use the actual password file and set inappropriate permissions. Then a hacker can easily get the password file as follows:

```
ftp target
Connected to target.
220 target FTP server ready.
Name (target:hacker):  anonymous
331 Guest login ok, send ident as password.
Password: ******
230 Guest login ok, access restriction apply.
ftp>  get /etc/passwd
```

The messages from the FTP server can vary; however, the numeric values are standardized on all implementations.

Proper configuration of FTP is a must. Misconfigured anonymous FTP can open your system to both the theft and destruction of data. Anonymous FTP should be configured as follows:

The anonymous FTP account entry in the system's password file, `/etc/passwd`, should be similar to

```
ftp:DISABLED:500:500::/users/anon_ftp:/bin/false
```

The entry in the system's group file, `/etc/group`, and login group file, `/etc/logingroup`, should be similar to

```
ftp:DISABLED:500:
```

If the permissions of the FTP directory tree are not configured correctly, a hacker may be able to add an executable file to the bin directory or put a .rhosts file in the account's home directory and allow himself remote access via the Berkeley Trusted Hosts commands.

The anonymous FTP home directory and its subdirectories should have ownership and permissions as follows:

```
drwxr-x---  6  root  ftp  512 Jan 1 08:00 .
drwxr-x---  9  root  bin  512 Jan 1 08:00 ..
drwxr-x---  2  root  ftp  512 Jan 1 08:00 bin
drwxr-x---  2  root  ftp  512 Jan 1 08:00 etc
drwx-wx---  2  root  ftp  512 Jan 1 08:00 incoming
drwxr-x---  2  root  ftp  512 Jan 1 08:00 pub
```

If any of these directories is owned by `ftp`, then an intruder could compromise the system.

The optional incoming directory allows anonymous FTP users to store files on the system. Anonymous FTP is allowed write permission into the incoming directory. This directory has only write and execute permissions, allowing FTP to write into, but preventing FTP from listing, the contents of the directory. Having an incoming directory will allow an anonymous user to fill up your disk space.

FTP's password and group files should contain no information other than the lines for FTP, as in the following examples:

FTP's password file:

```
ftp:DISABLED:500:500:::
```

FTP's group file:

```
ftp:DISABLED:500:
```

Any commands in the bin directory should have the permissions --x--x--x so no one can interrogate the binary or replace it.

Trivial FTP, TFTP, is a file transfer program that requires no authentication. No user name or password is requested. This program will send and receive files to or from anyone who asks for them. The program is generally restricted to send only those files that are in the home directory tree of the user named "tftp." TFTP is generally used to boot up network devices such as network terminal servers, network-based printers, and X terminals.

If you do not require TFTP, disable or remove the TFTP daemon, tftpd.

If your system has network access, you should try to get the password file with tftp, because hackers will.

```
$      tftp
tftp>  connect target.com
tftp>  get /etc/passwd /tmp/target.passwd
```

Hackers may also be able to place a .rhosts file in the tftp home directory and then use trusted services to gain access to your system. Be sure the secure flag is set on the tftpd daemon in the internet configuration file, and there is a tftp user with a properly secured home directory.

This flag will limit TFTP access to the tftp user's home directory. The entry in the internet configuration file should be similar to this:

```
tftp dgram udp wait root /etc/tftpd tftpd -s /u/tftp
```

On some systems the secure flag may be "-r". Consult the manual page for TFTP. If your system does not have a secure flag, then try the above commands. If you can get your password file, so can a hacker. If this is the case, contact your vendor for a secure implementation of TFTP.

Since the tftpd daemon is started by inetd, you can use inetd.sec to restrict access to your TFTP services to specific machines.

Services Which Enable Trust

Trust is the situation where one system will trust the security of another system and relinquish control of security to that system. Usually trust is limited to user authentication, so that any user from a trusted system is considered to be trustworthy and no further authentication will take place. This is a user convenience giving users single-login, but it opens the door to hackers. Once one system is compromised, all the other systems in the chain of trust are directly vulnerable.

Berkeley Trusted Hosts is a subsystem that allows global authentication on a group of trusted hosts or equivalent systems. These systems are said to trust each other. Specifically, this means you can have access to all the trusted hosts without having to reenter your password. It also means that if a trusted system is compromised, then all the systems that trust the compromised system must be considered compromised.

Generally, all systems in a trusted host group are similarly managed, often by the same administrator. This is where the hacker is most likely to start to expand his realm. The `/etc/hosts.equiv` file will show him all the trusted hosts. The trusted hosts will generally all have the same user IDs. The .rhosts files will show him systems that users have added as trusted hosts; these are especially useful to the hacker; since they were added by users and not the system manager, the systems that are being trusted may not have the same level of security. The ".rhosts" files allow a user to define another user ID on another system to use the current system as the current user. Trusted systems are generally reciprocally trusted, so if a user has a .rhosts file on this system allowing access from another system, it is likely that the other system has a similar .rhosts file.

To find the home directories that have a `.rhosts` file use the following script:

```
cut -d: -f6 /etc/passwd | xargs -i ls -l {}/.rhosts 2>/dev/null
```

This command is the basis for any tool that needs to locate or process files in a user's home directory. The cut command reads the sixth field of the password file, which is the home directory entry, and then uses xargs to execute a command for each user's home directory.

Trusted telnet is a facility which allows telnet to utilize the Berkeley trusted hosts facilities, so one can log in to a trusted host without using a password. A typical configuration may consist of one or more secure front-end systems and a network of participating hosts. Users who have successfully logged onto the front-end system may telnet directly to any participating system without being prompted for another login. This option supports the TAC User ID (also known as the TAC Access Control System, or TACACS User ID), and uses the same files as rlogin to verify participating systems and authorized users, hosts.equiv, and .rhosts.

Enable the TAC User ID option. The system administrator can enable the TAC User ID option on servers designated as participating hosts by having inetd start telnetd with the -t option in `/etc/inetd.conf`:

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd -t
```

In order for the TAC User ID option to work as specified, the system administrator must assign to all authorized users of the option the same login name and unique user ID (UUID) on every participating system to which they are allowed TAC User ID access. These same UUIDs should not be assigned to non-authorized users.

Users cannot use the feature on systems where their local and remote UUIDs differ, but they can always use the normal telnet login sequence. Also, there may be a potential security breach where a user with one UUID may be able to gain entry to participating systems and accounts where that UUID is assigned to someone else, unless the above restrictions are followed.

Syslogd, the system logging daemon, can receive and store log information from other systems. There is no authentication required to send data to this service, so that any information sent to the syslog daemon will be written in the log files. On Linux systems, the remote logging feature has to be enabled. However, on HP-UX, it is enabled by default and has to be disabled with the "-N" option to disable this anonymous access.

File System Access

Remote file system access is a common service for networked computers. Many proprietary systems developed elaborate file-sharing protocols. Remote file sharing creates a significant security problem. Access and authorization must be uniformly enforced across a number of remote systems. This requires a consistent administration of systems and users. However, the use of these remote file system protocols does not require the enforcement of this level of homogenous administration. So there are many security issues with the use of remote file systems.

NFS

NFS streamlines file sharing between server and client systems by controlling access via the `/etc/exports` file. Entries in `/etc/exports` provide permission to mount a file system existing on the server onto any client machine or a specified list of machines. Once a file system is put into `/etc/exports`, the information is potentially available to anyone who can do an NFS mount. Thus, the NFS client user can access a server file system without having logged into the server system.

- **Server Vulnerability** — Server security is maintained by setting restrictive permissions on the file `/etc/exports`. Root privileges are not maintained across NFS. Thus, having root privileges on a client system does not provide you with special access to the server.

  The server performs the same permission checking remotely for the client as it does locally for its own users. The server side controls access to server files by the client by comparing the user ID and group ID of the client, which it receives via the network, with the user ID and group ID of the server file. Checking occurs within the kernel.

  A user with privileges on an NFS client can exploit that privilege to obtain unlimited access to an NFS server. Never export any file system to a node on which privilege is granted more leniently than from your own node's policy!

- **Client Vulnerability** — In earlier releases of NFS for workstations, the `/dev` inode had to reside on the client's disk. NFS now allows for the /dev inode containing the major and minor numbers of a client-mounted device to exist on the server side. This opens the possibility for someone to create a Trojan horse that overrides permissions set on the client's mounted device, by accessing the device via the file and inode number found on the server side.

  Although lacking permission to make a device file on the client side, a system violator wanting to sabotage the client can create an undermining device file, such as `/dev/kmem`, using root permissions on the server side. The new `/dev` file is created with the same major and minor number as that of the target device on the client side, but with the following permissions: crw-rw-rw-.

  The violator can then go to the client, log in as an ordinary user, and, using NFS, open up the newly created server-side device file and use it for devious means — to wipe out kernel memory on the server, read the contents of everyone's processes, or other mischief.

Common Internet File System

The Common Internet File System, CIFS, is the Windows specification for remote file access. CIFS had its beginnings in the networking protocols, sometimes called Server Message Block (SMB) protocols, that were developed in the late 1980s for PCs to share files over the then nascent Local Area Network technologies (e.g., Ethernet). SMB is the native file-sharing protocol in the Microsoft operating systems and the standard way that millions of PC users share files across corporate intranets.

CIFS is simply a renaming of SMB, and CIFS and SMB are, for all practical purposes, one and the same. (Microsoft now emphasizes the use of CIFS, although references to SMB still occur.) CIFS is also widely available on UNIX systems, VMS™, Macintosh, and other platforms.

Despite its name, CIFS is not actually a file system unto itself. More accurately, CIFS is a remote file access protocol; it provides access to files on remote systems. It sits on top of and works with the file systems of its host systems. CIFS defines both a server and a client: the CIFS client is used to access files on a CIFS server.

Samba, the Linux implementation of SMB, is supplied as part of the standard package on most Linux implementations and on HP-UX.

The HP-UX implementation of CIFS is CIFS/9000. It enables directories from HP-UX servers to be mounted onto Windows machines and vice versa. The HP CIFS/9000 server product consists of Samba source code which has been enhanced with ACL (Access Control List) mapping features. These mapping features allow you to change UNIX permission bits and ACLs from an NT client through the NT ACL graphical interface on NT clients.

SMB is a very noisy protocol. Servers announce their presence and identify their shared file systems to anyone who asks. This allows the "network neighborhood" functions to be "user friendly." Servers can be configured to not broadcast their existence, but anonymous access to share names is available to anyone who knows the server name.

UUCP

UNIX to UNIX Copy (UUCP) is a utility that is designed to facilitate transferring files, executing commands on remote systems, and sending mail over serial dial-up lines. There are two versions of UUCP: Version 2 that was written in 1977 at Bell Laboratories and is running on some older systems and the more common HoneyDanBer UUCP which was released in 1983 with UNIX System V Release 3. It is easy to tell which it is by looking in the UUCP directory, `/usr/lib/uucp`. If there is a file named Permissions, then it is HoneyDanBer; if there is a file named USERFILE, then it is Version 2.

The security of both versions of UUCP can be increased by creating different accounts with their own unique user names and passwords for each system that will be calling your system. Each account will have to have the same user ID as the uucp account. This gives you more accountability. You can tell when each system logs in and out and you can disable a specific machine by disabling the account.

There are a number of configuration files that identify with which other computers to communicate and what permissions those computers have on your system. These files must contain appropriate configuration information and be properly protected.

The Systems file (L.sys in Version 2) contains the system name, phone number, uucp login name, and password for systems that the system calls. Even if the permissions are set correctly on this file, a hacker can get into this file by using the uuname command to get a list of systems that are called by this system, and using the debug option of cu or uucico to determine the phone numbers and uucp logins and passwords.

The Systems file should be owned by the account "uucp" and be readable only by uucp. The debug options for cu and uucico should be disabled if possible; otherwise, the command uucp should be executable only by the account "uucp," and cu removed from the system, unless needed.

In Version 2, the L.cmds file contains a list of commands that can be executed by the specified remote system. All unnecessary commands should be removed.

The USERFILE is used to set local access permissions. It identifies for each system what directories that system has access to. It also will indicate if dial-back is to be utilized for the system.

The directories that can be accessed should be restricted. You should not allow access to any user's home directory or any directory that contains configuration information. Altering this information could compromise your system.

The HoneyDanBer system combines the functionality of these two files into one file, the Permissions file. The Permissions file is made up of a number of name/value pairs. Each line will define the accessible directories and available commands for a MACHINE that your system calls or a LOGNAME for a system that calls your system.

A hacker may try UUCP access to the system. If your system supports anonymous UUCP, this will let him browse. You should try the following hacker trick to make sure your system is secure:

```
uucp  target!/etc/passwd /tmp/target.passwd
```

If the target system has not limited the scope of the file system access, this will get the password file. A machine that is not properly configured may allow a hacker to update the system's configurations.

Today, most sites have replaced their use of UUCP for point-to-point access with SLIP or PPP, a point-to-point networking protocol. However, most systems still have the UUCP software loaded.

If you have UUCP on your system, whether you use it or not and whether you have modems or not, you must validate that the Permissions file is configured appropriately. An inappropriately configured UUCP can be used to gain privileges on the local system. If you are not using UUCP, remove it from your system.