AIX 4.3 Elements of Security

Effective and Efficient Implementation

- Achieve confidentiality, integrity, and availability
- Practical examples and best practice recommendations
- Gain insight into AIX security management

Yoshimichi Kosuge
Francois Armingaud
Lip-Ping Chew
Leonie Horne
Timothy Witteveen

Redbooks

ibm.com/redbooks

SG24-5962-00

International Technical Support Organization

AIX 4.3 Elements of Security Effective and Efficient Implementation

August 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special notices” on page 211.

First Edition (August 2000)

This edition applies to AIX Version 4 Release 3 (5765-C34).

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Contents

Preface

This IBM Redbook provides an overview of AIX Version 4.3 security. AIX provides many security features that can be used to improve your system security.

This document does not replace any other published documents. It is intended as an additional source of security information and, together with existing sources, may be used to enhance your knowledge of security. The emphasis is on the practical use of these security features, why they are necessary, and how they can be used in your environment. We also provide examples and best practice recommendations.

The reader is assumed to have a basic working knowledge of UNIX. This book is intended for experienced AIX system administrators who are taking on the role of security administration. Security administrators who are new to AIX will also find this document useful.

This document uses the redbook *Elements of Security: AIX 4.1*, GG24-4433 as a base and reflects what has changed since AIX 4.1.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Kosuge Yoshimichi is an IBM RS/6000 SP project leader at the International Technical Support Organization, Poughkeepsie Center. Since he joined IBM, he has worked in the following areas: LSI design, S/390 CP microcode, VM, MVS, OS/2, and AIX. After joining the ITSO in 1998, he has been involved in writing redbooks and teaching IBM classes worldwide on all areas of the RS/6000 SP system.

Francois Armingaud is an I/T Specialist and Certified AIX Systems Administrator in France. He has 22 years of experience in IBM, 15 in the AIX field. He holds a degree in Computer Science from Ecole des Mines de Nancy (1971). He taught Computer Science there and also in Ecole Polytechnique of Algiers, Centre d'Etudes et Recherches en Informatique in Oued Smar, and Ecole des Mines de Paris, of which he was a member of the educating board. Francois then worked for IBM with major customers in France including Alcatel, Bouygues, Dassault, Matra, Poclain, Thomson. His areas of expertise include AIX and Linux system administration, coordination

between UNIX and Windows 9x/NT/2000 systems, and application development. He was for five years, from 1995 to 1999, the system administrator of IBM ECAM (European Center for Applied Mathematics) in Paris, France.

Lip-Ping Chew is the Lead Technologist in StarSecure, Singapore. He has over four years of experience in the security field. He has worked at Reuters and Netcentre, providing security and e-commerce consulting. His areas of expertise include Internet and operating systems security. He holds a computer science degree from the National University of Singapore. He is a Certified Information Systems Security Professional (CISSP).

Leonie Horne is an I/T Specialist from IBM Sydney, Australia. She joined IBM Australia in 1990. For the past two years, Leonie has worked in the AIX technical support area of Information Technology Services (ITS) and is a certified IBM AIX v4.3 System Administrator.

Timothy Witteveen is a Senior Research Scientist at Pacific Northwest National Laboratory. Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy. He has over four years of experience in SP System Administration. He is a certified IBM AIX v4.3 System Administrator.

Thanks to the following people for their invaluable contributions to this project:

William R. Ogden
International Technical Support Organization, Poughkeepsie Center

Lee Terrell
Scott Trent
IBM Austin

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 249 to the fax number shown on the form.
- Use the online evaluation form found at `ibm.com/redbooks`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Introduction

This book is about AIX 4.3 security. AIX provides many security features that can be used to improve security. The emphasis is on the practical use of these security features, why they are necessary, and how they can be used in your environment. We also recommend guidelines and best practices when there are many different ways to achieve a secure system.

This introduction provides a high-level overview of basic security concepts and different approaches of achieving security. It is important to recognize that AIX security is an integral part of achieving security. At the same time, simply implementing the AIX security features without considering other facets of security can result in systemic vulnerability.

The rest of this book covers different aspects of security present in AIX 4.3. This includes user accounts, file systems, networks, and security management. Finally, it provides you with security products information.

This document is intended for experienced AIX system administrators who are taking on the role of security administration. Security administrators who are new to AIX will also find this document useful.

IBM produces many publications that deal with the actual syntax and configuration of AIX security. Readers are advised to consult the following technical documentation:

- *AIX Version 4.3 System Management Concepts: Operating System and Devices*, SC23-4311
- *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127
- *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126
- *AIX Version 4.3 System User's Guide: Operating System and Devices*, SC23-4121
- *AIX Version 4.3 System User's Guide: Communications and Networks*, SC23-4122
- *AIX Version 4.3 Files Reference*, SC23-4168

1.1 Why is security important?

The “2000 Computer Security Institute/Federal Bureau of Investigations Computer Crime and Security Survey” (`http://www.gocsi.com/`) states that 90 percent of respondents detected computer security breaches within the last 12 months and 74 percent acknowledged financial losses due to computer breaches.

The “American Society for Industrial Security/PricewaterhouseCoopers Trends in Proprietary Information Loss Survey Report 1999” (`http://www.pwcglobal.com/`) states that Fortune 1000 companies sustained losses of more than $45 billion from thefts of their proprietary information.

The exponential growth of networks has caused more and more computers to be connected together. This growing inter-connectivity and easy access to anyone with a PC creates an excellent environment for information exchange and sharing. Unfortunately, this very property also increases the risk of information being leaked or stolen. As an increasingly large amount of confidential information is stored or transmitted over public networks, such as the Internet, it becomes imperative that information security be implemented in an effective and efficient manner.

Therefore, what security aims to achieve is:

Protection of assets
: The individual bits and bytes may define either important corporate or national security information. The modification or leakage of such information may cause huge financial losses or even loss to human life.

Protection from liability
: If due professional care is not exercised, an organization might be liable for losses caused by poor computer security.

Protection of brand
: Branding and reputation has always been important. The protection and projection of an image or reputation is crucial to most organizations. As an example, most users would prefer to purchase online from a well-known site rather than a little known one.

Protection from stoppage
: The downtime of a computer caused by a denial-of-service (DOS) attack causes loss of service and revenues.

Protection of morale

The inability to perform normal work functions or the loss of work due to a break-in can potentially impact the morale of the work force. This usually causes a vicious cycle of events where users may even abandon present security measures because they were not sufficient.

Protection of stock price

The widespread distributed denial-of-service (DDOS) attacks on February 2000 has shown that stock prices can be affected by computer security problems.

1.1.1 Threats to security

There is no need to lock our doors if there are no thieves. Unfortunately, this is not the case in the physical or virtual world. We need security to protect our assets. It is useful to take a look at the threats that our computers are up against. Threats can come from both internal and external sources. Far too often, companies setup firewalls when connecting to the Internet and forget about the security of backend servers. This creates a tough outer shell but a fragile inner core. A disgruntled employee may simply Telnet into one of these backend servers and cause considerable damage. Also, it is possible for someone to write a compiler that it inserts a backdoor when it compiles login programs, or when it compiles other compilers. See *Reflections on Trusting Trust*, by Ken Thompson at `http://www.acm.org/classics/sep95/`. You have to trust someone, but know who you trust.

This document emphasizes on the need for good security at the host level.

The following are major threats to computer security:

Human errors

A keying error may delete the wrong file, for example. Secured files and a good backup procedure (which is also an element of security) reduce these problems.

Disgruntled employees or ex-employees

These may be after confidential information or may wish to sabotage the system.

Curiosity

For example, everyone would like to read (and perhaps change) payroll and personnel files. Breaking into computers has also become an ego trip for many teenagers (otherwise known as script-kiddies). For example, some of them could run scripts that

others created to break into systems without a true understanding or ability to generate/identify their own security holes.

Industrial espionage
The ability to obtain intelligence information about another country or company is part of an overall goal to achieve competitive advantage for commercial or military purposes.

Ideological or political statements
This is an increasingly common occurrence on the Internet. The government Web sites of US, Taiwan, Japan, and Indonesia have all been hacked in to.

Commercial goals
Both employees and external hackers are tempted by financial end-goals. The ability to extort or transfer large amounts of money is attractive to many people.

1.1.2 Goals of security

Computer security goals are usually defined in terms of:

Confidentiality Assets are accessible only by authorized parties.

Integrity Assets can only be modified by authorized parties in authorized ways.

Availability Assets are accessible to the authorized parties.

Before achieving the preceding goals, information needs to be classified. Information that is open to the public does not need confidentiality but needs integrity if it is used to make important decisions. An example is stock quotes or other financial information.

The security measures that can be taken to achieve the security goals are varied and multi-faceted. In 1.2, “Approaches to security” on page 7, we look at security from different layers: physical, network, operating system, and applications. While there are different measures that can be taken at each layer, they all share certain security principles:

Principle of least privilege
Run both users and applications with the least possible privilege. There is no need to allow either an application or user to have higher privileges than necessary. An unintentional user mistake may be just as serious as a malicious act on the computer. An application running with high privileges may potentially modify or read sensitive files for passwords and permissions.

Principle of minimum access

Only allow what is needed and deny all others. This is probably the most important security principle to keep in mind. By not running unnecessary services, past and future vulnerabilities do not apply anymore. As an example, if sendmail is not run, then all vulnerabilities of sendmail can be ignored.

Security in depth

Do not depend on only one security measure for defense. Instead make use of a variety of measures from the different layers to complement each other. See 1.2, “Approaches to security” on page 7. And when a security measure fails, it fails safely by disallowing access.

Legal liabilities

Comply with local and international laws. Some of the legal aspects to take note of include software licensing, import and export of cryptographic products, data privacy, and computer misuse laws. Consult your legal department.

Always keep these principles in mind when making decisions.

1.1.3 Adequate security

No matter how much security you implement, it will not be effective unless it is implemented properly and administrated regularly. If it is not implemented properly, it will either interfere with the users ability to do work, or it will not stop a hacker. Increasing system security requires increasing administrative time and effort.

Smaller organizations may simply let the administrators handle it intuitively as part of a systems administration process. This approach tends to be more tactical and dependent on the capability of the administrators.

Large organizations may have security or audit teams working in tandem with the administrators or developers. This approach is methodical and strategic in nature. Naturally, the required resources are more in this case.

Whichever the case, the most important aspect is to have proactive security. To plan for security when something means that a loss has occurred. Stop gap measures are seldom effective over the long run and may introduce problems of their own.

One way is to determine the cost of implementing the security measures versus the assets that are being protected. Unfortunately, assets such as work force morale and reputation are difficult to quantify. We recommend that

security decisions are not made solely by the technical department but also involve other stakeholders, such as the business and legal departments. Ultimately, the decision on how much security is one of trade-off.

As security increases, so does cost as shown in Figure 1. Cost represents time, resources, and opportunity costs while security is being implemented, where the cost-benefit cross-over point depends largely on the circumstances and company policies. One hundred percent security is a myth in the networked world. But to ignore security until a security breach happens is worse. The loss of assets and reputation is far greater than what most security solutions cost in the first place. Prevention is better than the cure.

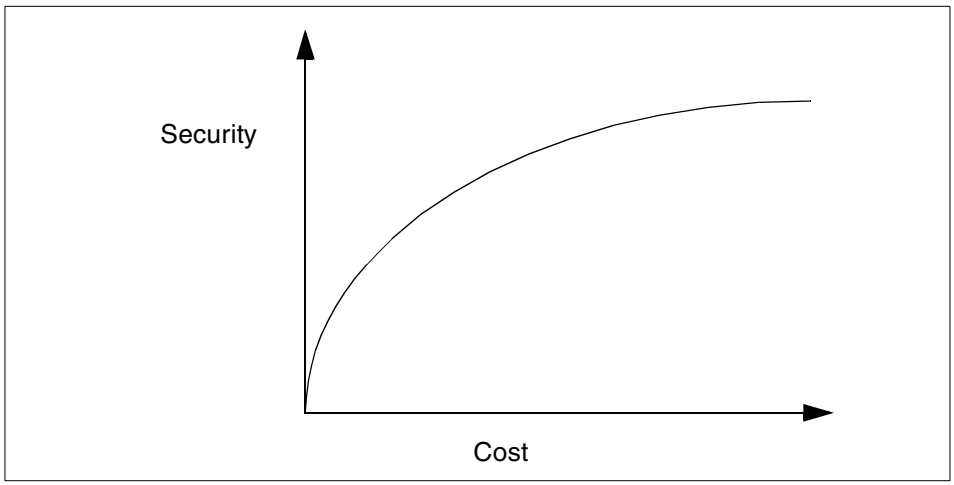

Figure 1. Security versus cost

The risk management process is useful for coming up with a picture of what is at risk and the action that can be taken. The objective is to have acceptable risks given the circumstances. In Figure 2 on page 7, the threats are identified, associated risks listed, and risk mitigation is performed. We recommend that risk management be used as part of your security process.

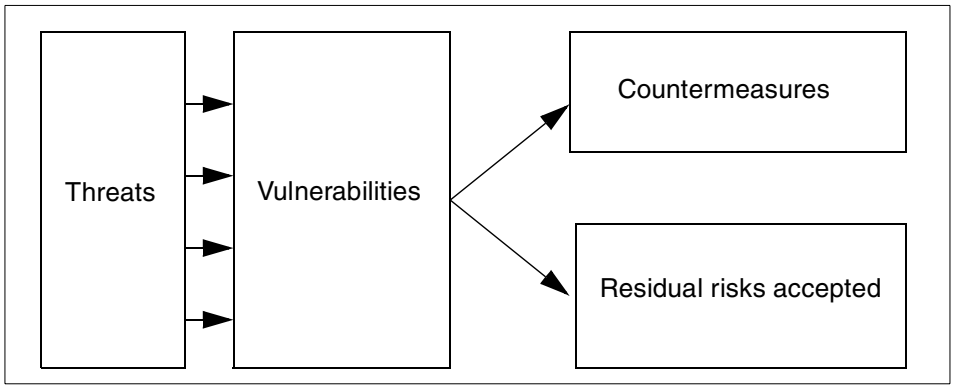

Figure 2. Risk management process

We do not go deeper into the risk management process, but it is useful to note that the security market for products and solutions have increased considerably in the past few years. The commercial adoption of cryptography products, firewalls, intrusion detection systems, and virtual private networks have helped Information Technology (IT) management contain the risks in a cost-effective manner. We recommend the adoption of security technologies as effective countermeasures to the risks. Security consultants and organizations have also increased to cope with the increasing demands placed on security that is currently beyond the reach of conventional Management Information System (MIS) setups.

This document is an example of a typical risk countermeasure. By implementing the guidelines outlined in this document, you are able to reduce the risks that come with deploying an AIX system on your network.

1.2 Approaches to security

Security at an operating system level is only part of a security framework. It is important to recognize this because security cannot be achieved piecemeal.

Figure 3 on page 8 shows a typical security framework. This book covers the security aspects of AIX 4.3. The remainder of this section covers the other pieces of security.

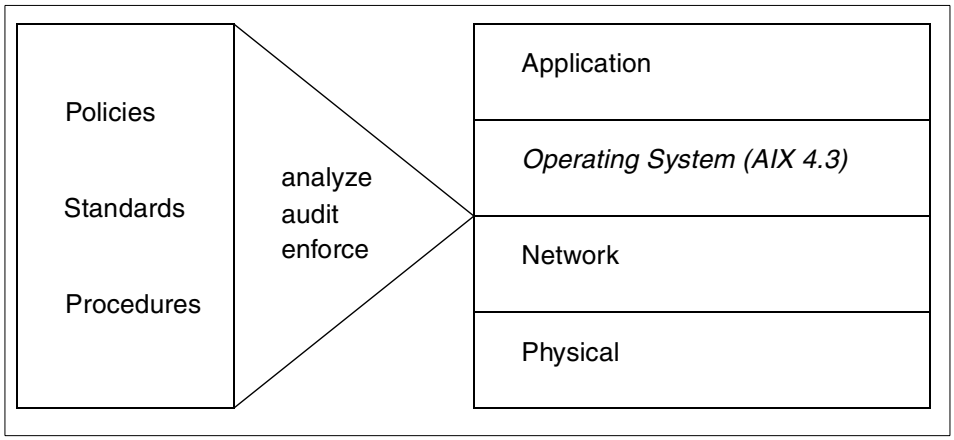

Figure 3. Simple security framework

The physical network, operating system, and applications all work together to achieve total security. The policies, standards, and procedures provide the necessary conditions.

1.2.1 Policies, standards, and procedures

Policies, standards, and procedures are the backbone of your infrastructure security framework. Smaller organizations tend to ignore or add to policies on an as-needed basis. This is not advisable and defeats the purpose of security policies.

Policies define the high-level objectives, while the standards and procedures deal with the tactical issues. As an example, a policy states that weekly backups must be performed on your system. The standards or procedures define which day of the week and the commands to perform for the backup. Figure 4 on page 9 shows the hierarchical structure between policies, standards, and procedures. The rest of this section makes use of “policy” to encompass all three terms (policies, standards, and procedures).

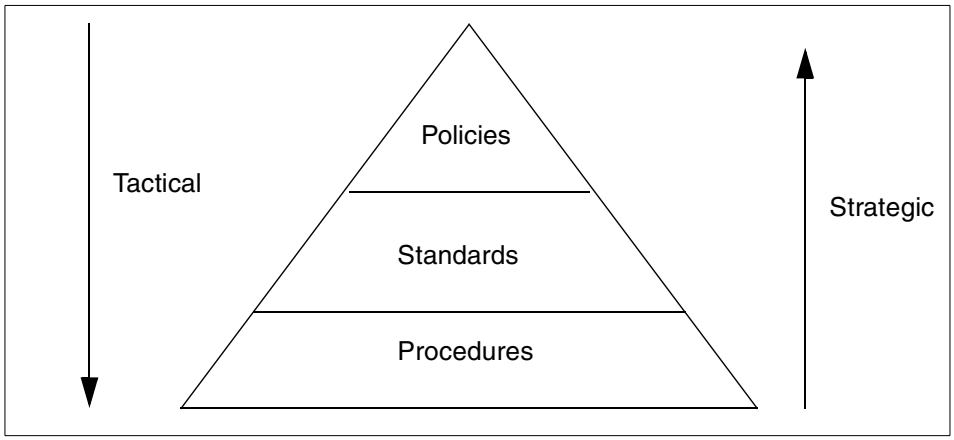

Figure 4. Policies, standards, and procedures

In general, policies have the following advantages:

Fair treatment
Policies ensure that there is fair treatment. A policy ensures that everyone is aware how a certain item is to be handled.

Liability abatement
The existence of policies shows that due care has been taken to address security issues.

Correct and consistent approach
Policies provide a correct and consistent approach. This prevents ad-hoc approaches that are detrimental to overall security.

Team consensus
Much of the value in policies comes from the policy creation process where different functional groups come to a consensus on the way things need to be done.

Knowledge transfer
The existence of policies ensures that new employees can readily get up to speed on how things are done or expected in the organization.

1.2.2 Physical security

Physical security is important. The most common way to implement security is to enforce physical access control to a physical site. Practically, this may not be entirely effective due to the large number of outsiders and contractors

that have physical access to a typical site. The way that data cables are laid may also be constrained by the physical properties of the building.

We must emphasize, however, the fundamental exposures of physical access to a communications channel. The exposures include:

Ethernet promiscuous mode
: In general, many Ethernet (and IEEE 802.3) adapters provide a method of monitoring all traffic on their LAN. Many TCP/IP packages provide a separate module to conveniently use this function. Monitoring can be from any location on the LAN. Anyone with a small PC, the appropriate software, and a connection to the LAN can monitor all data traffic on the LAN. The LAN connection might be an established one (at a connector on an office wall, for example) or an actual “tap” in the LAN cable.

Token-ring performance and monitoring adapter
: This is a standard IBM product that can display all the traffic on a LAN. It can be used from any location on the LAN. (In principle, this adapter can be purchased by anyone, but, in practice, it is not widely available.)

LAN analyzers
: These can display all traffic on a LAN. They can be used from any location on the LAN.

Data scopes for SNA/SDLC/HDLC lines
: In general, these require RS-232 (or similar) interfaces and are used only at modem locations. It is possible to “tap” a line with a receive-only modem arrangement and use a data scope for monitoring the line.

ASCII start-stop monitoring
: Many common PC modems can operate in a receive-only monitoring mode. This can be used to “tap” a dial-in session anywhere the base band telephone signal can be found, such in local telephone wires, wiring closets, and so forth.

Newer LAN topologies, using various types of switches, have the potential for much better security using modes in which session traffic is seen only by the sending and receiving nodes. We recommend the use of Virtual LANs, sticky ports, and flood control on your switches. The use of security measures, such as encryption, at the network layer also overcomes the physical exposures on the network.

When wireless LANs are used in placed of physical cables, we recommend the use of wireless spread spectrum technology. Spread spectrum was

invented by the military to ensure integrity and secrecy of wireless transmissions. The two most common spread spectrum implementations are Direct Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS). Both techniques prevent casual eavesdropping on the traffic. Both DSSS and FHSS are widely available on current wireless LAN products. Certain vendors also add in security features, such as encryption and manual acceptance of client MAC addresses. These are useful if a higher level of security is required.

Another aspect of wireless security are those used by mobile service networks. With the convergence of data and voice, the mobile phone has become a “computer”. Messages are sent to the hand phones directly. Currently, we do not recommend that confidential messages be sent. The Wireless Transport Layer Security (WTLS) provides authentication, privacy, and integrity for the Wireless Application Protocol (WAP). The adoption of WAP as part of mobile services is growing but not truly established yet.

IBM RS/6000 hardware security
AIX 4.3 runs on IBM RS/6000 exclusively. Using AIX security features in conjunction with IBM RS/6000 hardware security features, you can improve your system security.

IBM RS/6000 provides the following three hardware (including firmware) security features: cover lock key, power-on password, and privileged-access password. For more information refer to section 4.5, “IBM RS/6000 hardware passwords” on page 68.

1.2.3 Network security

A well-designed network has the ability to protect segments of machines at one time. This is in contrast to host security where individual machines have to be secured. The most common form of security on the network is the firewall. Firewalls are choke points between different networks. Because all traffic passes through this point, they can provide an effective form of access control on content and traffic type. A typical firewall design is shown in Figure 5 on page 12. Note that this is only an example. All the traffic to and from the Internet (untrusted network) is through the firewall. The network can be further segregated into a trusted internal network and the Internet servers.

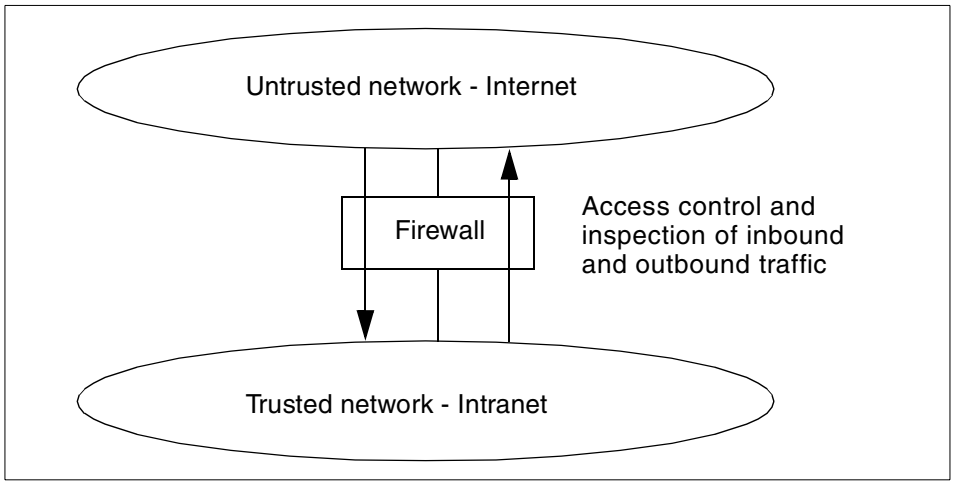

Figure 5. Example deployment of firewalls in a corporate network

The use of firewalls and good network design is integral to any good security deployment. But the other factors in the security framework have to be taken care of as well to ensure there are no loopholes and to add depth in security.

The use of virtual private networks (VPN) has increased in popularity over the last few years. AIX provides IP Security (IPSec) capability for tunneling data packets in a secure channel over untrusted networks. IPSec has been adopted in a wide variety of networking products, such as routers, firewalls, and dedicated VPN boxes. We recommend testing the interoperability between different products (especially between different vendors) before making any purchase. For more information on IP Security, refer to 7.3, “IP Security” on page 136.

Intrusion Detection Systems (IDS) is the complementary partner to firewalls. IDS is like burglar alarm. It notifies users of intrusions and provide a good depository of network audit information. IDS can be used to detect internal misuse or breaches of security policy. For more information on IDS, refer to 8.3, “Intrusion detection and recovery” on page 179.

1.2.4 Application security

Application refers to any software that has been installed on the AIX. The number one rule in application security is not to install anything that is not needed. This security stance should always be kept in mind even when installing new AIX machines. There is absolutely no need to install the numerous file sets that are on the AIX CD-ROMs.

The following are points to take note of in terms of application security:

Ownership and permissions

Occasionally, some applications install with poor permissions or do not take into account of your specific setup. The essential question is: Does someone or some group require access to the application or parts of it? Do not run as root, if possible, because a stack buffer overflow can allow root access immediately.

Logging

Turn on logging when possible. This provides an audit of the usage of the application. The flip side is the maintenance and monitoring of the logs.

Trusted source

Do not use software from untrusted sources. When software is signed, always verify the signature with a public key from a trusted Certificate Authority (CA) or keyserver.

Compilers and interpreters

Compilers and interpreters are useful but they add significant risks by providing an additional avenue for exploit.

setuid/setgid properties

Certain applications require setuid/setgid. This is potentially dangerous if the programs did not practice safe coding techniques. Again, if setuid/setgid are not required, change it.

Use security features

Many applications provide security enhancements. Examples of security features include virtual roots, IP access control, and cryptographic security. These security features are there for a reason. Be sure to understand and use them.

Transient files

Certain applications create temporary files with poor permissions. These files may be overwritten with a Trojan Horse. These files are usually in the /tmp or application directory.

Demand for security

Commercial products depend on customers for revenue. Demand security if products do not have or have insufficient security.

Chapter 2. Users

AIX supports the standard user attributes usually found in the /etc/passwd and /etc/group files, such as:

Authentication information
This specifies the password. The password is discussed in Chapter 4, “Passwords” on page 57.

Credentials
This specifies the user ID (UID), principal group ID (GID), and the supplementary GID. GIDs are discussed in Chapter 3, “Groups” on page 45.

Environment
This specifies the home or shell environment.

AIX allows for greater control, if desired, with extended attributes. Security information can also be separately protected from public access.

Note that this chapter does not cover the effect of NIS and NIS+ usage. The following discussion assumes administration is performed directly on the system. NIS is discussed briefly in 7.6, “Network information system (NIS)” on page 159.

2.1 User IDs

A user has two forms of identification in basic UNIX systems. A user may have more than two if additions, such as DCE, CICS, and other subsystems, are considered. They are a *user name*, such as joe, and a *user ID* (*UID*). For example, user name joe may have UID 201. All internal system identification uses the UID. The only place where the user name is used is the /etc/passwd file and it is closely associated administration files, such as /etc/group and the shadow and configuration files in the /etc/security directory.

Every user should have a unique UID. If SMIT is used to create new users, it automatically gives every user a unique UID. Although you can override this by forcing a UID number in SMIT or by editing the /etc/passwd file, such actions should be done with caution. If two users have the same UID, they are the same user for all practical purposes. A reasonable security policy should prohibit assigning the same UID to multiple individuals.

2.1.1 Sharing a UID

You may encounter some arguments for sharing UIDs. This usually occurs in the context of a particular application. The argument goes like this:

- Each user sharing one UID must still login under their own user name and must enter a unique password. This removes any need to share passwords.
- The application and its data files can be made more secure by limiting access to only the owner, and the owner is the shared UID. This allows only user names with the shared UID to access the files.
- The application program, by internal design and processing, provides all necessary security and can differentiate between the various user names sharing a UID.

The problem with this logic is that other applications do not differentiate between all the user names sharing a UID. Accountability is lost. The desired level of sharing can usually be accomplished with unique UIDs and appropriate GIDs and usage. Nevertheless, there may be existing applications that require shared UIDs.

Another example for the use of shared UIDs is for training. In this case, many users may know the password of a user training account. Obviously, these UIDs should be restricted to specialized groups, have very restrictive quotas, and so forth. The best suggestion is to limit training accounts to specific systems. Without using special shells, it is difficult to rigorously limit a UNIX account, and you may not want user training to take place on your production systems.

You can check for shared UIDs by displaying the /etc/passwd file. For example[1]:

```
joe:*:201:1::/home/joe:/usr/bin/ksh
jane:*:202:1::/home/jane:/usr/bin/ksh
```

The third operand of each line is the UID. If two users (two lines) contain the same number, then they are sharing a UID. The fourth operand is a group ID (GID), and it is normal for multiple users to share GIDs. If a considerable number of users exist, checking UIDs manually could be tedious. The `usrck` command is available to do these checks and is described in 3.3, “Verifying the user environment” on page 54.

[1] If AIX has been configured to use LDAP or DCE as the user security repository, all of the users will not be in the /etc/passwd file.

2.2 System-defined users

As distributed, AIX has users and groups that are needed by the system. Do not alter these users and groups unless you are very certain about what you are doing. Do not log in to any of these users (except root).

The following users are supplied with the system (in the form in which they appear in /etc/passwd). They are used for various purposes, such as file ownership and NFS functions. All except root have been disabled for login in the distributed system. They are disabled by password = * in /etc/security/passwd. They are:

```
root:!:0:0::/:/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/home/guest:
nobody:!:4294967294:4294967294::/:
lpd:!:9:4294967294::/:
```

New users that you add default into the *staff* group, unless you change the default. A larger multi-user system should have locally-defined groups. You may find the *system* and *security* groups useful if you have multiple administrators, since these groups allow execution of many administrative functions. Refer to 3.1.5, “System-defined groups” on page 49 for information on groups.

2.3 The root user

All UNIX systems have a *super user*. By convention, this user is named *root*. It is the UID = 0 that provides this user with all system administration functions.

Many user and system administration functions require you to be root. This is because:

- Root owns the files being updated.
- Root can write in the files regardless of who owns them.

For practical purposes, a system administrator must be able to run as root; the primary system administrator of AIX is the root user. For personnel backup, maintaining multiple shift operations, and so forth on larger systems, there probably are several people who know root’s password. This disclosure

of a password is probably necessary but must be managed with considerable care.

In AIX 4.2, the function of *roles* was introduced. Roles consist of authorizations that allow a user to execute functions normally requiring root user permissions. Refer to 3.2, “Administrative roles” on page 50 for more information.

It is possible to share root’s UID; that is, have multiple login user names that equate to UID = 0. This permits several administrators, each with their own password, to have root authority. This situation is not uncommon, but we recommend against it. Among other problems, it encourages routine operation as root.

Where multiple administrators are required (each of whom needs to have root authority for certain functions), we strongly recommend that each administrator log in with their own user name (which has its own unique UID) and then `su` to root when required. For the `su` command, refer to 2.3.1, “The su command” on page 18.

We have two strong recommendations for root:

- Protect the password. Your system has no protection against anyone with root’s password.
- Do not routinely operate as root. Switch to it only when needed for an administrative action. This prevents accidents, which are the source of many security failures in UNIX.

2.3.1 The su command

The `su` command allows a user to temporarily switch to another user’s identity provided the user knows the second (that is, switched to) user’s password or the first user is the root user. If `su` is used with no operand, it attempts to switch to the root. The `su` command requires the user to enter the password of the target user. Therefore, anyone obtaining root authority through `su` must know root’s password.

AIX keeps a log of all uses of the `su` command. This log is in /var/adm/sulog. This log can be useful for an overview of how `su` is being used, or for reconstructing a series of events. It logs all use of `su`, including switches to root. Refer to 8.1, “Accounting and logs” on page 171 for a discussion of AIX logs.

2.3.2 The PATH environmental variable

The PATH is an environment variable used by the current shell when searching for executable files (commands). When using a normal shell, a user can change their PATH specification at any time. There is no reasonable way to prevent changes. The restricted Korn shell, discussed in 2.5.3, “Restricted Korn shell” on page 33, does not permit changes to PATH.

One security goal is to prevent root (or any other user for that matter) from executing a counterfeit program (a Trojan Horse). For example, if /tmp (an unprotected directory) is the first element in PATH, and if someone places a program named su in /tmp, then this su is executed instead of the correct system su program[2]. The PATH exposure is a simple concept, and the system administrator must understand it. You should devote whatever time is necessary to understand it. You cannot hope to maintain a secure system if you do not understand PATH handling.

A user’s PATH is normally set (using the system profile and the user’s profile if it exists) when logging into the system. Both /etc/profile and $HOME/.profile are executed automatically when a user logs into the system. The root user often has the root directory as the home directory, and /.profile (if it exists) is executed in addition to /etc/environment when root logs into the system. The biggest difference is that the default root’s PATH does not include current directory (.). If you want to be really safe, you should not include dot (.) in your root’s PATH, even at the end.

When switching user identities with the `su` command, the target user’s profile is not automatically executed. Using a “`-`” flag with the `su` command causes the target user’s profile to be executed, but this may have after effects on the current user after exiting from the target user’s identity. Typically, the “-” flag is not used with `su`. For example, if you log in as a normal user and then `su` to root, you continue to use the profile (and PATH) established by the original user identity. This can be the source of serious exposures, similar to this:

1. A user (wanting to obtain root’s password) writes a small C program to counterfeit the initial appearance of the `su` command. That is, it asks for a password.
2. The user compiles and links this program into their home library.
3. The user alters the PATH to search their home directory first, before looking in various system directories.
4. The user asks the administrator for help with a problem that is likely to require root access.

[2] To be safe, administrators should invoke the fully-qualified path name for the commands. For example, use `/usr/bin/su` instead of `su`.

5. The administrator sits at the user's workstation and uses `su` to switch to root. When the administrator enters the `su` command, the system searches the current home directory (as directed by the PATH) and finds the counterfeit `su` program and executes it.
6. The counterfeit program prompts the administrator for the root password, stores the password in a hidden file, sends an error message indicating an incorrect password, and erases itself.
7. The administrator thinks she/he has entered the wrong password and tries again. This time, the correct `su` command obtains control (because the counterfeit program is gone) and the session continues normally.
8. The user later reads the root password from the hidden file and is able to login as root.

This is the classic Trojan Horse attack and it worked because the administrator executed `su` using the wrong PATH. There are two lessons to be taken from this type of attack:

- Administrators, when executing as root, should always enter the full path name of commands if they are working under another user's environment. This avoids usage of the existing PATH definition.
- The PATH for a normal user should search the standard system directories before searching the current directory or specific $HOME directories.

The default PATH (set by the default user profile) for AIX is:

```
/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.
```

Subdirectories within /usr contain most of the AIX commands used by normal users. The /etc directory contains symbolic links to commands in more remote directories. Notice that the system libraries are searched first. After the system libraries are searched, $HOME/bin (a conventional location for products installed in the user's home directory) is searched. The "dot" in the last position of the PATH is significant. This indicates that the current directory should be searched.

Ignoring minor elements (X11 and /sbin), the PATH search order for this PATH is system directories, home directory (bin), and current directory. This is a fairly safe search order.

2.3.3 Disabling root user

There is seldom a good reason for logging in as root. Most system accidents in UNIX are partly caused by routine use of root as a working user. After your system is installed, you may want to disable the ability to login as root.

Authorized users (those who know the password for root) could then `su` to root after they login under their normal user names.

Disabling root is easily done with SMIT. Use the `smit chuser` fast path. After selecting the root user, change the fields as follows:

```
smit chuser
Another user can SU TO USER?                    true
User can LOGIN?                                 false
User can LOGIN REMOTELY?                        false
```

Do not disable root by editing /etc/passwd and changing the password field. This also prevents you from using root through `su` or `telnet`, which means root will not be able to log into the system. If this happens, refer to 2.3.4, “Repairing root user” on page 21.

2.3.4 Repairing root user

If something goes wrong with root, you have a serious problem. Forgetting the root password is a serious problem. New system administrators, while experimenting with various security options, may make the system so secure that no one can use it. To “break into” the system in order to repair it:

1. Boot your system into a limited function maintenance shell (Service or Maintenance mode) from bootable AIX media.

   Please refer to your system user's or installation and service guide for specific IPL procedures related to your type and model of RS/6000. Additionally, the document titled “Booting in Service Mode”, available at `http://techsupport.services.ibm.com/rs6k/techbrowse/`, has specific procedures for most types of RS/6000s.

2. With bootable media of the same version and level as the system, boot the system into Service mode.

   The bootable media can be any ONE of the following:

   - Bootable CD-ROM
   - NON_AUTOINSTALL, bootable mksysb
   - Bootable Install Tape

3. Select language from the displayed menu.
4. Select **Start Maintenance Mode for System Recovery** from the displayed menu.
5. Select **Access a Root Volume Group**.
6. Select **Continue** and respond to any additional prompts.

7. Select **Access the Volume Group and start a shell**. When this completes, you have normal access to all your system files, and you are executing as root.
8. You may need to set TERM and EXPORT TERM in order to use your display in full-screen mode.
9. Use SMIT or other commands to repair your system.
10. Do the `sync` command to ensure that the disk has been updated.
11. Remove the CD-ROM from the CD-ROM drive and perform `shutdown -Fr` when you have finished.

This procedure assumes that the CD-ROM drive is included in your boot device sequence and it is placed before hard disk. If this is not the case, you have to change your boot device sequence using System Management Services (SMS).

It seems that anyone can use this method of “breaking into” the system if they have the installation CD-ROM (and it is available anywhere). To avoid this situation, use one or both of the following password setting:

1. Set the power-on password. The power-on password protection is effective against reset as well as power-on. The power-on password will be required to boot your system. Refer 4.5, “IBM RS/6000 hardware passwords” on page 68.
2. Leave only hard disk in the boot device sequence, and set the privileged-access password. The system will boot only from hard disk. The privileged-access password will be required to add the CD-ROM drive to the boot device sequence. Refer 4.5, “IBM RS/6000 hardware passwords” on page 68.
3. Put the machine in a physically secured room.

2.3.5 Single-user systems

Some degree of security is important even for a single-user system. At a minimum, you probably want to prevent the wrong people from using the system. Default users (such as root) without passwords are part of the basic AIX system. Anyone with physical or network access to the system can use it until you change these defaults.

Note

The root password should be created immediately after installation of the AIX operating system. This is set through the “Configuration Installation Assistant” menu immediately after the system boots for the first time.

Most single-user systems are networked, and no system on a network is really single-user. Remember that being attached to a network is a two-way relationship. See Chapter 7, “Networks” on page 123 for more information. It is foolish to ignore security in any workstation connected to a network.

Even if your system is not used by anyone other than yourself, do not use root as your routine user. Many security checks are also accident prevention checks. Operating as root often overrides the accident-prevention aspects of security. As an extreme example, perhaps you want to display /etc/passwd, but you absent-mindedly enter `rm /etc/passwd` instead of `pg /etc/passwd`. If you are not root, you receive an error message. If you are root, you have created a substantial problem for yourself.

2.4 User

User administration can be done via SMIT or by editing files or using mid-level commands such as `mkuser`, `chuser`, or `rmuser`. The following section refers to commands through SMIT whenever possible. Modern UNIX systems, such as AIX, have more files involved in user administration than earlier UNIX systems and using SMIT helps ensure consistent updates to the files.

You may want to edit /etc/security/.profile before adding users. This is the prototype for the user’s own profile; this file is copied to $HOME/.profile as part of the SMIT process of creating a new user. Users can change their own $HOME/.profile file, if they know how. The prototype is used only when a new user is created.

Note

.profile file is for the Korn shell and Bourne shell while .login file is for C shell. .login is extracted from /usr/lib/security/mkuser.sys file.

Give some thought to user names, since they are used by humans. Employee numbers, for example, make poor user names because they are not meaningful to other people. e-mail is important on most systems and

meaningful user names make handling mail much easier. Users are much happier if their user name is the same on all systems they use. This is easy if the new user already has an established user name in the organization. If the user is new to the organization, some coordination effort may be required to assign a user name. An organization's security standards should contain guidelines for creating user names and associating these with UIDs.

2.4.1 User parameters in SMIT

The following menu is produced by the `smit mkuser` fast path to add a user. Subsets of these same menu items are produced by other SMIT operations.

```
1 User NAME                                          []
2 User ID                                            []
3 ADMINISTRATIVE USER?                               false
4 Primary GROUP                                      []
5 Group SET                                          []
6 ADMINISTRATIVE GROUPS                              []
7 ROLES                                              []
8 Another user can SU TO USER?                       true
9 SU GROUPS                                          [ALL]
10 HOME directory                                    []
11 Initial PROGRAM                                   []
12 User INFORMATION                                  []
13 EXPIRATION date (MMDDhhmmyy)                      [0]
14 Is this user ACCOUNT LOCKED?                      false
15 User can LOGIN?                                   true
16 User can LOGIN REMOTELY?                          true
17 Allowed LOGIN TIMES                               []
18 Number of FAILED LOGINS before                    [0]
   user account is locked
19 Login AUTHENTICATION GRAMMAR                      [compat]
20 Valid TTYs                                        [ALL]
21 Days to WARN USER before password expires         [0]
22 Password CHECK METHODS                            []
23 Password DICTIONARY FILES                         []
24 NUMBER OF PASSWORDS before reuse                  [0]
25 WEEKS before password reuse                       [0]
26 Weeks between password EXPIRATION and LOCKOUT
27 Password MAX. AGE
28 Password MIN. AGE
29 Password MIN. LENGTH
30 Password MIN. ALPHA characters
31 Password MIN. OTHER characters
32 Password MAX. REPEATED characters
33 Password MIN. DIFFERENT characters
34 Password REGISTRY
```

```
35 Soft FILE size
36 Soft CPU time
37 Soft DATA segment
38 Soft STACK size
39 Soft CORE file size                          [2097151]
40 Hard FILE size                              []
41 Hard CPU time                               []
42 Hard DATA segment                           []
43 Hard STACK size                             []
44 Hard CORE file size                         []
45 File creation UMASK                         [022]
46 AUDIT classes                               []
47 TRUSTED PATH?                                nosak
48 PRIMARY authentication method               [SYSTEM]
49 SECONDARY authentication method             [NONE]
```

The line numbers (1 to 49) are not shown in the SMIT menu but are shown here to assist the following discussion. Some of these options are important for security and must be understood.

Line 1: User NAME

This field is the user name for the new user. The user name must be unique on a given system.

Line 2: User ID

This field is the UID. The SMIT process automatically assigns the next UID; you should not override this field without a very good reason. Leave it blank.

Line 3: ADMINISTRATIVE USER?

Indicates the administrative status of the user. Only the root user can alter the attributes of an administrative user.

Line 4: Primary GROUP

Specifies the group of the user when the user first logs in. If you do not enter or list any groups, the system assigns the user to the primary default group specified in the /usr/lib/security/mkuser.default file.

Line 5: Group SET

Specifies groups to which a user is a member. A user can be a member of up to 32 groups.

Line 6: ADMINISTRATIVE GROUPS

Specifies the non-administrative groups for which the user is an administrator.

Line 7: ROLES

Allow non-root users to be assigned portions of root privileges for

system administration purposes (Refer 3.2.4, “Role authorizations” on page 53).

Line 8: Another user can SU TO USER?

Determines whether any other user can switch to this account by using the `su` command. The default is set to true. This field is not effective against root, who can `su` to the account regardless of user restrictions.

Line 9: SU GROUPS

Provides some control over which other users can `su` to this user (if Another user can SU TO USER? is set to true). By entering the group name, or if multiple groups, then the group name followed by a comma, only allows access to this account if members of the named group. An exclamation mark preceding a group name functions as a “NOT” symbol, meaning members of the named group are not permitted to `su` to this user. This field is not effective against root who can `su` to the user regardless of group restrictions.

Line 10: HOME directory

Specifies the full path name for the user’s home directory. When the user logs in, the system sets the current directory to be the user's home directory. If you do not enter a home directory, the system creates the home directory using the default specified in the /usr/lib/security/mkuser.default file and stores the path in the /etc/passwd file.

Line 11: Initial PROGRAM

The full path name of the program to be given control when this user logs into the system. It is normally the name of a shell, such as /usr/bin/ksh. If this field is left blank, the initial program name is taken from /usr/lib/security/mkuser.default.

Line 12: User INFORMATION

Contains general comments about the user. For example, you can list the user's full name, department, employee serial number, or office location.

Line 13: EXPIRATION date (MMDDhhmmyy)

Usually left as 0 (meaning no expiration date). The date format is shown in the menu. A typical entry might be “0630000001” (MMDDhhmmyy). This parameter is useful for temporary accounts, such as for visitors or contractors. The account is disabled after the specified date.

Line 14: Is this user ACCOUNT LOCKED?

This field is normally false and can be used as the single point of control to disable a user. This does not force a user off the system if the user is currently logged in.

Line 15: User can LOGON?

Determines whether this user can login directly. A normal user should be allowed to log into the system. However, standard system users, such as bin, are special cases in which no login should be permitted. This parameter sets a flag in the /etc/security/user stanza relating to this user, as well as an asterisk in the password field in /etc/passwd to inhibit login.

Line 16: User can LOGIN REMOTELY?

Controls access through the `rlogin` and `telnet` functions. If set to true (the default), anyone (anywhere on a connected network) can log into this account using Telnet if they know the user's password. This parameter does not control access through the FTP function.

Line 17: Allowed LOGIN TIMES

Limits a user to logging in between certain hours. There are several formats for this parameter, and they are explained in the comments in /etc/security/user. Once logged into the system, a user cannot be forced off if their session extends beyond the valid hours.

Line 18: Number of FAILED LOGINS before user account is locked

Number of consecutive unsuccessful login attempts the user is allowed. If this number is exceeded, the account is locked and the user cannot login. If 0 is specified this feature is disabled.

Line 19: Login AUTHENTICATION GRAMMAR

This field should be left with the default "compat" value. This specifies the methods through which the user must authenticate successfully before gaining access to the system. The default value compat indicates that normal log in procedures is followed. Therefore, compat allows local and NIS users access to the system (this is defined through the SYSTEM attribute in /etc/security/user).

Line 20: Valid TTYs

Defines the terminals this account can use. It does not control pseudo-terminals as used by Telnet and other remote connections. The full path name of terminals must be given, such as /dev/tty1. An exclamation mark before a terminal name means that terminal may not be used. The ALL value means that all terminals may be used. The ability to limit specific users to specific terminals can be

a strong security tool, but must be used with care. For example, root might be limited to the local console, although this leaves an exposure in the event of a hardware problem.

Lines 21 to 33:
These are various password controls and are discussed in Chapter 4, “Passwords” on page 57.

Line 34: Password REGISTRY
Do not enter anything here. This specifies the authentication mechanism through which the user is administered. It is used to resolve a remotely administered user to the locally administered domain. This situation might occur when network services unexpectedly fail or network databases are replicated locally.

Lines 35 to 44: The process limitations
Provide some protection against runaway programs. Limits are categorized as either soft or hard. With the `ulimit` command, you can change your soft limits (lines 35 to 39) up to the maximum set by the hard limits (lines 40 to 44). You must have root authority to change resource hard limits. Refer to 5.4.1, “ulimit” on page 95.

Line 45: File creation UMASK
Default umask for the user. A user can change their umask value, for the duration of a session, with the `umask` command. Refer to 5.3, “umask” on page 94.

Line 46: AUDIT classes
Specifies the audit classes to trace while the user is working on the system if auditing is enabled. Audit classes are a collection of audit events (such as events generated when the user logs in or when a user attribute is changed) determined by the administrator.

Line 47: TRUSTED PATH?
Specifies a variable that defines the user's access to the trusted path. The system uses this variable when the user tries to invoke the trusted shell or a trusted process, or enters the secure attention key (SAK) sequence (refer to Chapter 6, “Trusted computing base (TCB)” on page 109).

Line 48: PRIMARY authentication method
User authentication is used to control access to an RS/6000. A primary authentication method will prevent login access. If you leave this field blank, the system uses the method set in the /etc/security/user file (refer to 4.4.1, “Additional authentication - auth1 and auth2” on page 65).

Line 49: SECONDARY authentication method

Specifies a second level of identity and access privilege authentication that the system should use if the user successfully logs in to this user account. The system recognizes the same values as used in the primary authentication but is purley informational and will not prevent login access. If you leave this field blank, the system uses the method set in the /etc/security/user file.

Files associated with new users

The results of adding a new user (using the SMIT) are additions to several files shown in Table 1.

Table 1. Files associated with new users

File name	Description
/etc/passwd	Contains a new line defining the user.
/etc/security/passwd	Contains a new stanza for the user's encrypted password and a few flags. Updated with a new user once a password has been set.
/etc/security/user	Contains a new stanza containing some of the user's restrictions.
/etc/group	Altered to add the user to one or more groups.
/etc/security/limits	Contains a new stanza containing some of the user's environmental limits if other than the default values.
/etc/security/.ids	This is updated to contain the next available UID.
/home	Contains a new directory, which is the home directory for the user (this assumes normal definitions in the SMIT).

Refer to 2.6, "Files associated with user accounts" on page 37 for a detailed description of the preceding files.

2.4.2 Setting user system defaults

The complete set of user attributes is defined in:

- /usr/lib/security/mkuser.default
- /etc/security/user
- /etc/security/limits
- /etc/security/login.cfg
- /usr/lib/security/mkuser.sys

Many of the SMIT menu fields described in 2.4.1, “User parameters in SMIT” on page 24 might be best set as default values instead of individual user values. For example, you might want to establish a value of Password MAX. AGE for all users rather than setting it for each user.

To change most defaults, you need to edit the /etc/security/user file. There is no SMIT menu to alter the default stanza. All other user stanzas can be changed by SMIT. There is a stanza for every user defined in the system and a default stanza in this file. As root or a member of the security group, you can change the default stanza using the `chsec` command or by editing the /etc/security/user file directly. Changes are effective immediately, the next time a user logs into the system, unless that user has overriding values in their own stanza. Also, parameters in the default stanza appear as default values the next time you add a user with SMIT.

Of the 49 menu items displayed by the SMIT add/change user menus, 30 are stored in /etc/security/user and can be controlled by values in the default stanza.

The /etc/security/limits file is organized in the same way, and defaults for the 11 user limits (soft file size, and so forth) can be set here.

The /etc/security/login.cfg file also has default settings that can be changed for all users.

The /usr/lib/security/mkuser.default file is useful for making a generic change to all users’ primary groups, shell, and home directories.

The /usr/lib/security/mkuser.sys file is the script that creates the user’s home directory and copies to the appropriate login scripts (C shell or Korn/Bourne shell).

The advantages of setting default values instead of many individual user values are obvious. The default values can be changed easily. Individual user values should be specified only when they need to be different than the default values. The /etc/security/user and /etc/security/limits files are used whenever a user logs into the system; changes do not affect a user who is currently logged into the system.

2.4.3 Disabling users

On occasion, it may be necessary to disable a user. For example, dormant user accounts present a particular security exposure. These are accounts not in regular use and, therefore, are not regularly monitored for signs of attempted entry. Some accounts that are considered dangerous in that they

are common to UNIX platforms are guest, ftp, mail, bin, and adm. While some of these accounts are required to enable system functions, they all should have a password and/or be locked and any account that is dormant should be disabled.

To disable an account, use the smit security fast path:

```
User NAME                                          joe
User can LOGIN?                                    false
User can LOGIN REMOTELY?                           false
Valid TTYs                                        []
Allowed LOGIN TIMES                               []
Is this user ACCOUNT LOCKED?                       false
Number of FAILED LOGINS before                  [0]
user account is locked
Login AUTHENTICATION GRAMMAR                      []
```

To lock an account, use the smit user fast path:

```
* User NAME                                         joe
Is this user ACCOUNT LOCKED?                      true +
```

To remove a user, use smit rmuser fast path:

```
User NAME                                         [Joe]
Remove AUTHENTICATION information?                 yes
```

The Remove Authentication information field indicates if the system should delete the user's password and other user authentication information from the /etc/security/passwd file.

When removing a user through SMIT, it does not remove the users files or their home directory. The existing files become “unowned”. This creates a security exposure if the files are not removed.

The `find` command can be used to list all files owned by a specific user (before the user is deleted). For example:

```
find / -user joe
```

We recommend running this command before removing a user from the system. These files can then be checked, and useful ones allocated to other users by using the `chown` command. The remainder can then be deleted.

Refer to 5.1.5, “Unowned files” on page 80 for more information about unowned files.

2.5 Controlling a user's environment

AIX provides various ways of allowing a security administrator to control a user's environment as discussed in the following sections.

2.5.1 Timeouts - Automatic logoff

All systems are vulnerable if terminals are left logged in and unattended. The most serious problem occurs when a system administrator leaves a terminal unattended that has been enabled with root authority. In general, users should log off any time they leave their terminals.

Timeouts are used to automatically log out a terminal that has been inactive too long. The Korn shell uses the TMOUT and the Bourne shell uses the TIMEOUT parameters in the /etc/profile file. You (the administrator) should set one or both of these variables if you want to automatically log off terminals after an excessive idle period. We strongly recommend using this function because unattended terminals are serious security exposures.

The following example forces the terminal to log off after an hour of inactivity:

```
TMOUT=3600
TIMEOUT=3600
export TMOUT TIMEOUT
```

Note

Users can override the TMOUT and TIMEOUT values in the /etc/profile file by specifying different values in their *.profile* file of their home directory. This can be prevented by specifying the `readonly` command. For example:

```
TMOUT=3600
readonly TMOUT
export TMOUT
```

Alternately, you could place the timeout variable in /etc/security/.profile (from where it is copied to the home profile of new users). The timeout is expressed in seconds. A reasonable timeout value can be the subject of much argument, but probably is between 10 and 120 minutes.

The timeout period is for the shell. If a user nests several shells (by issuing the `ksh` command repeatedly, for example), the shells timeout in reverse order with each one taking the full timeout period.

2.5.2 Locking your terminal

You can lock your terminal with the `lock` command. The `lock` command requests a password from the user, reads it, and requests the password a second time to verify it. In the interim, the command locks the terminal and does not relinquish it until the password is received the second time. The timeout default value is 15 minutes, but this can be changed with the `-number` flag.

If your interface is AIXwindows, use the `xlock` or `xss` command in the same manner.

For example:

- To reserve a terminal under password control, enter:

  ```
  lock
  ```

  You are prompted for the password twice so the system can verify it. You receive a message advising that the “session will timeout in 15 minutes.”

- To reserve a terminal under password control with a timeout interval of 10 minutes, enter:

  ```
  lock -10
  ```

  It is a good idea to encourage users to adopt this practice prior to leaving their terminals for an extended period of time, for example, going to lunch, and so on.

2.5.3 Restricted Korn shell

The restricted shell is used to set up login names and execution environments whose capabilities are more controlled than those of the regular Korn shell. The `ksh -r` command opens the restricted shell. The behavior of this command is identical to those of the `ksh` command except that the following actions are not allowed:

- Cannot change the current directory (`cd`).
- Cannot change the value of the SHELL, ENV, or PATH variables.
- Cannot invoke a program by specifying the explicit path name of that program.
- Cannot redirect output with >, >|, <>, >>.

A basic user should have little, if any, requirement to perform the preceding functions. A restricted shell is a great security tool.

The restricted shell is set up by editing the user's /home/.profile or by changing the user's default shell using the `smit user` fast path.

Suggested steps to setting up a restricted shell:

1. Create the restricted Korn shell:

   ```
   ln -s /usr/bin/ksh /usr/bin/rksh
   ```

2. Add /usr/bin/rksh to the "usw" entry of /etc/security/login.cfg.

3. Change the users default shell (for example, user joe):

   ```
   chsh joe /usr/bin/rksh
   ```

   Or use SMIT as follows:

   ```
   User NAME                                [joe]
   Initial PROGRAM                          [/usr/bin/rksh]
   ```

4. Prevent the user from modifying their .profile:

   ```
   cd /home/joe
   chown bin:bin .profile .
   chmod -w .profile .
   ```

If a user working in a restricted Korn shell invokes a Korn shell script, the script runs normally. That is, a restricted Korn shell does not restrict Korn shell scripts. In this way, it is possible to provide an end user with shell procedures that access the full power of the Korn shell while imposing a limited menu of commands.

Security tips:

- Be aware if a user can write to their home directory, they can force overwrite their .profile file and get out of the restricted shell.
- Ensure the user's PATH is set to one directory that does not include any shell programs (such as `csh`, `bash`, `tcsh`, and so forth). The user can simply run one of these and be free from your control.
- It is a good idea to have a /usr/local/rbin directory and set the user's PATH to that. Then, you can control what programs are available to the user by placing them in that location.
- You can launch a menu program from the .profile so they never see a command line.

2.5.4 Assigning security controls to ports

Security controls can be assigned to specific ports (such as /dev/ttyp0). These controls are stored in /etc/security/login.cfg, which is discussed in 2.6,

"Files associated with user accounts" on page 37. To do this, use the `smit login_port` fast path. After selecting a port, you see the following SMIT menu:

```
* Port NAME                                    /dev/ttyp0
  Allowed LOGIN TIMES                         []
  Login RETRY DELAY                           [0]
  Number of FAILED LOGINS before              [0]
  port is locked
  INTERVAL for counting failed logins         [0]
  REENABLE DELAY for locked port              [0]
```

The exact format and meaning of these parameters is documented in the comments in /etc/security/login.cfg. The combination of the RETRY DELAY, FAILED LOGINS, INTERVAL, and REENABLE DELAY can provide good protection against repeated attacks on a dial-up port in which the attacker is attempting to guess a user name or password.

The /etc/security/login.cfg file contains a default stanza and, potentially, a stanza for each port. We suggest using the default stanza rather than specifying values for each port (unless you need different values for different ports, of course). Unlike system user defaults (see 2.4.2, "Setting user system defaults" on page 29), the port default values may be set using SMIT by entering "default" as the port name.

2.5.5 Displaying a user's current directory

You may want to set the shell prompt to show the current directory. This helps users to know which directory they are in at all times. For Korn shell users, this is done by adding the following two lines to the users $HOME/.profile:

```
PS1='$PWD $' (use single quotes)
export PS1
```

Alternately, you could add this to the /etc/profile, /etc/security/profile, or /etc/environment files so that any new users created will automatically have the PS1 attribute set to show their current directory.

This change provides a shell prompt with the current path name followed by the traditional "$." Unfortunately, this simple technique provides a misleading prompt if the user executes a `su` to root. The "$" will still appear instead of the "#", which is traditional for root. This can be avoided two ways:

1. Not using this alteration for users who frequently `su` to root.
2. Using the command `su -` (with the "-" flag) when changing to root.

A prompt displaying the current directory path is helpful for many users, especially if they routinely work with multiple directories. There are no

security elements involved (other than the misleading prompt after `su` to root), but a more informative prompt may reduce user errors.

If you (the system administrator) are working on multiple machines over a network, it may be useful to show the machine name of the current machine you are working on as well as the current directory. This can be achieved by adding the following line to root's .profile file:

```
PS1="[`hostname -s`]["'$PWD]# '
export PS1
```

There are many more sophisticated methods for obtaining informative prompts. Almost any book discussing UNIX usage or administration will contain suggestions for shell prompts. We recommend the simple version (listed here) only if it fits your needs or if you are not comfortable with (that is, do not understand) more complex prompt functions described elsewhere.

2.5.6 Obtaining information before login

A potential intruder may be able to obtain important information about your system even before login, such as host name and version/release of AIX.

To prevent giving away important information, you can edit the /etc/security/login.cfg file. The herald attribute defines the login message printed when the getty process opens the port. The default herald is the login prompt.

You can change this to add a message by running the `chsec` command or by editing the /etc/security/login.cfg file directly.

The following is an example if you use the `chsec` command:

```
chsec -f /etc/security/login.cfg -s default -a herald=" NOTICE TO
USERS\r\n\r\nUse of this machine waives all rights to your
privacy,\r\n\r and is consent to be monitored.\r\n\rUnauthorized use
prohibited.\r\n\r\n\r\nlogin: "
```

The following is an example if you edit the /etc/security/login.cfg file directly:

```
default:
herald =" NOTICE TO USERS\r\n\r\nUse of this machine waives all rights
to your privacy,\r\n\r and is consent to be monitored.\r\n\rUnauthorized
use prohibited.\r\n\r\n\r\nlogin: "
        sak_enabled = false
        logintimes =
        logindisable = 0
        logininterval = 0
        loginreenable = 0
```

```
logindelay = 0
```

The Common Desktop Environment (CDE) can also be customized so as to not reveal information. This is achieved by the following:

1. Copy /usr/dt/config/$LANG/Xresources to /etc/dt/config/($LANG)/Xresources.
2. Edit the Xresources file entry for greeting and change to the required text.

The logo can also be changed to a specified bitmap image.

2.6 Files associated with user accounts

The files that are associated with user administration are listed here with brief comments. Almost all files directly associated with user and group administration are in the /etc/security directory.

/etc/security/.ids

Do not edit this file. It contains the sequence numbers the `mkuser` command uses so that a new group or user always gets a unique UID/GID. The file is updated automatically by various commands invoked (internally) by SMIT. An example of this file is:

6 221 12 206

Where:

6 = next administrative UID number
221 = next UID number
12 = next administrative GID number
203 = next GID number

/etc/group

Contains basic group definitions. You would normally update this file through SMIT, but you may edit it directly. (The plus sign (+) beside an entry means to refer to the NIS server for additional entries. Never use the “+” unless you are certain NIS is installed and available in your network.)

/etc/security/group

Contains additional group information, such as *adms* and *admin* flags. You would normally update this file through SMIT, but you may edit it directly.

/etc/security/login.cfg

Contains stanzas for a variety of system-wide controls. There are no per-user stanzas. The controls are generally related to terminal

and port usage. Some of the parameters can be set through SMIT (refer to 2.5.4, “Assigning security controls to ports” on page 34), while others can be changed only by directly editing this file. Comments in the file describe the functions and formats very clearly. Stanzas include:

herald

This specifies the initial screen display before a user logs in. You may edit and redesign the herald display. Be certain your herald contains enough new-line characters to clear the screen. Refer to 2.5.6, “Obtaining information before login” on page 36.

Invalid logins

A group of parameters related to *invalid logins.* These parameters can delay or prohibit (for a period or indefinitely) additional logins after a failed login. These parameters can provide valuable protection for a system under attack by someone attempting to guess user names and/or passwords. If you have dial-in ports or are exposed to a large group of potential intruders, you should edit and use these parameters.

sak_enabled

This controls the availability of the secure attention function for a port. This is discussed in Chapter 6, “Trusted computing base (TCB)” on page 109.

auth_method

This defines different or additional authentication methods. See 4.4.1, “Additional authentication - auth1 and auth2” on page 65.

usw

This is used only by the `chsh` command and is a list of valid shells. Specify full path names. (The `chsh` command changes the initial program parameter in the user’s line in /etc/passwd.)

maxlogins

This sets the maximum number of direct terminal users who may be logged in at one time. (This parameter should be changed with the `chlicense` command, used in accordance with your AIX usage license.)

logintimeout

This specifies the time within a login must complete.

/etc/passwd

Contains basic user definitions. An exclamation point (!) in the password position is normal and causes the system to look in /etc/security/passwd where the encrypted password is kept. If the ! is replaced with an asterisk (*), the user is locked. The `passwd` command replaces the * with an ! while defining a password. (A plus sign (+) beside an entry indicates a switch to NIS for additional entries.) You would normally update this file through SMIT, but you may edit it directly.

/etc/security/passwd

Contains encrypted passwords, a time-stamp of the last update, and a flag indicating whether the password was updated by the administrator. (If so, the user is prompted to change the password the next time they log in.) You normally update this file through SMIT. You would edit it directly only in special circumstances, since you cannot directly enter the encrypted password.

/etc/passwd.dir and /etc/passwd.pag

Are created by the `mkpasswd /etc/passwd` command and contain small database structures to speed access to the user administration files. Do not edit these files.

/etc/security/user

Contains most of the user control parameters described in 2.4.1, “User parameters in SMIT” on page 24 and 2.4.2, “Setting user system defaults” on page 29. You would normally update this file through SMIT, but you may edit it directly. You should browse this file and become familiar with its format and contents.

/etc/security/environ

This is an ASCII file that contains stanzas with the environment attributes for individual users. You can specify exceptions from the default user environment defined in /etc/environment; for example, give a user a different NLSPATH (displaying messages in another language). You may edit this file directly. Each stanza is identified by a user name and contains attributes in the format:

```
Attribute=<Value>
```

/etc/security/limits

Contains soft resource parameters. These can be important on multi-user systems to prevent a single user from consuming too much of the system’s resources. There is one stanza per user and a default stanza. All of these parameters can be set through the user functions of SMIT. You can edit this file directly, but we

recommend using SMIT instead. Refer to 5.4.1, “ulimit” on page 95 for more information about limits.

/usr/lib/security/mkuser

Default contains some defaults used when creating a new user. You may edit this file directly. It contains the default group, the default initial program (shell), and the default home directory name for a new user.

/etc/security/failedlogin

Contains an entry for every time a login fails. The file can be displayed with the `who` command:

```
who -a /etc/security/failedlogin >> /tmp/check
```

This example redirects the output to a file called /tmp/check. Do not edit this file. However, after an extended period, you might want to delete it and allow the system to recreate it to recover disk space. This file is not as useful as it could be because it does not record invalid user names, that is, any user name that is not in your /etc/passwd file. Invalid user names are recorded as UNKNOWN rather than as the actual ID entered. Recording invalid user names is, itself, a potential security exposure, because entering a password when the system wants a user name is a common error.

/etc/security/lastlog

Has one stanza per user and contains information about several last logins (valid and invalid). Information from this file is displayed at user login time. You may display the file, but do not edit it. The timestamps in the file are unreadable by humans so that displaying it is of little value.

/etc/security/.profile

This is the prototype for the $HOME/.profile file for new users. You may edit this file and change it as required. It has no effect except when a new user is created.

/etc/profile

Provides a system-wide login profile for all users. Any user’s individual .profile (in their home directory) can override parameters in /etc/profile. You may edit this file directly. Typical contents include:

TMOUT/TIMEOUT

This defines the time (in minutes) that a user can be idle before they are automatically logged out of the system. TMOUT is used by `ksh` and TIMEOUT by `bsh`.

Refer to 2.5.1, “Timeouts - Automatic logoff” on page 32.

PATH

Local options are often included here. Examples are local PATH variables for product libraries, a call to /usr/games/fortune, and so forth.

/etc/security/audit/config

This is an ASCII stanza file that contains audit system configuration information.

2.6.1 Old files

Several of the files listed in the preceding section have a second, older, copy in the /etc/security directory. The “old” copy has the letter “o” as the first character of the file name. For example, /etc/security/limits and /etc/security/olimits both exist. The SMIT processes (that is, the lower-level commands called by SMIT) copy the current file to the “old” version for recovery purposes. This process is automatic. You should never touch the “old” files unless a disaster corrupts the operational files. The “old” files are normally one level down from the operational files; that is, the most recent change is not reflected in the old files.

2.6.2 Environment and variable files

When a user’s login is complete, their initial environmental and shell variables are from (in order):

1. /etc/environment
2. /etc/profile
3. /etc/security/environ
4. $HOME/.profile
5. $HOME/.kshrc

   A shell configuration file specified by the ENV environmental variable in $HOME/.profile.

6. $HOME/.Xdefaults

   If it exists and AIXwindows is used.

The first three files in this list are system files and must be protected. Improper environmental or shell variables can create many security holes. The last three items are owned by the user, and the user can alter them in

any way they wish. Only the owner should have write access to these files, and there is no real reason for anyone else to have read access.

The files /etc/environment and /etc/profile contain similar types of environment and profile information. Both files are executed for every user at login time.

The /etc/security/environ file can also contain similar environmental commands, but there is a stanza in this file for each user.

2.6.3 Shadow files

Shadow files have been introduced into AIX and other UNIX systems to create better security.

For example, the /etc/passwd file is used in a variety of ways. The lines in /etc/passwd are used by many programs to translate between a user name and UID. For this reason, /etc/passwd must be readable by any program and any user. That is, it must be “world readable”. For this reason, a shadow file called /etc/security/passwd is used to store the encrypted passwords for all users.

A line in /etc/passwd (which is readable by anyone) can have four types of entries in the password field (the second field in a line) for each user:

1. A null entry. (Fields are separated with colons.) A null password field means no password is required for this user.
2. An asterisk. This is one way to disable a user. (AIX normally uses another field in /etc/security/user to disable a user, but uses the asterisk method when a user is first defined.)
3. An exclamation mark. This means that the encrypted password is in the shadow file /etc/security/passwd. This is the normal case for AIX.
4. An encrypted password. An encrypted password is always 13 characters long.

Some accounts can have encrypted passwords in /etc/passwd, and other accounts can have their encrypted passwords in /etc/security/passwd. AIX works properly with both cases, but we strongly recommend against placing encrypted passwords in /etc/passwd since this is a security exposure. SMIT and the `passwd` command automatically place an exclamation mark in /etc/passwd and place the encrypted password in /etc/security/passwd. There are several other files in the /etc/security directory. These are all related to security controls and are sometimes called the “security shadow files.”

2.7 Important hints and tips

The following are hints and tips for administrating users:

- When assisting a user, try not to `su` to root from the user's session. If you do this, you are using their environment (with their own PATH), and this opens a large number of exposures. If you must do this, then use full path names for all commands you use while executing as root.
- Beware of a user who changes the IFS variable in their .profile. IFS is the Internal Field Separator and is used by programs, such as `cut`, `awk`, and `sed`. Do not allow it to be changed in /etc/profile. A knowledgeable user can do clever things with IFS and cause endless trouble.
- Beware of a user experimenting with ASCII terminal "tricks." It is possible to send control strings to many ASCII terminals to set up various function keys. A clever user can send, for example, a series of commands "hidden" with the normal operation of a function key. In the proper circumstances, such as when root uses the terminal, these "hidden" commands can be used to cause commands to be executed under the root UID without roots knowledge.
- The `who am i` command displays the login name associated with your terminal. This is unchanged by usage of the `su` command. The command `whoami` displays the current (effective) user name and changes when `su` is used. You normally want to use `whoami` and not `who am i`.
- Never place the current directory in the PATH for root. If you log into the system as root, AIX automatically removes the current directory (if it is specified) from the initial PATH. You can (but should not) restore it to the PATH after the login is complete.

Chapter 3. Groups

In larger installations, good system administration usually revolves around group definitions. Guidelines for forming groups should be part of any security policy. Defining groups for large systems can be quite complex and is beyond the scope of this document. Users will often belong to more than one group, but group membership should not be excessive. One goal is to have a moderate and stable number of groups. A symptom of poor security and poor administrative planning is a constantly increasing number of groups.

If at all possible, group definitions should extend across all system platforms: MVS, UNIX, NetWare, and so forth. That is, a given group name should have the same members, the same security associations, and similar administration on MVS and UNIX and LAN systems. Good group definitions are often related to job functions instead of strict organizational structures; for example, there may be a group for secretaries, regardless of their department. This is a difficult goal. System administrators will seldom do it voluntarily because it requires endless coordination and meetings with other system administrators. Nevertheless, it is a good goal because it forms and helps enforce a meaningful security policy for an enterprise.

The standard UNIX file security controls, the permission bits, provide very limited granularity. The AIX ACL functions extend this and are discussed in 5.6, “Access control list (ACL)” on page 100. Well planned use of group definitions substantially extends the usefulness of the permission bits.

However, it must be admitted that most UNIX administrators ignore group definitions, or, at best, define groups for use only within their system. It can be argued that no (or minimal) group definitions are better (more secure, less hassle) than poorly planned group definitions. Poorly planned groups tend to overlap in unexpected (and unsecure) ways, especially if a new group is defined for every new situation.

Our recommendations are:

- If possible, coordinate group definitions in as large a context as possible. At the enterprise level is best.
- At whatever level the groups are defined, think! Consider scenarios. Ask for advice. Once established and in use, group definitions are exceptionally difficult to change.

3.1 AIX group usage and administration

A group is a set of users, all of whom need access to a given set of files. Every user is a member of at least one group and can be a member of multiple groups.

Every user ID (UID) is assigned to a group with a unique group ID (GID). The system manager creates the groups of users when setting up the system. When a new file is created, the operating system assigns permissions to the UID that created it, to the GID containing the file owner, and to a group called others, consisting of all other users. The `id user` command shows your UID, GID, and the names of all groups you belong to.

There are three types of groups on the system:

- User groups
- System administrative groups
- System-defined groups

A user can be a member of multiple groups, and AIX will automatically search all of a user's groups for file access permissions. This allows good control with a reasonable number of groups, even for complex organizations.

In some cases, the multiple group search for permissions might lead to unanticipated results. If ACLs are used (5.6, "Access control list (ACL)" on page 100), it is possible that conflicting levels of authority exist for different groups.

The `newgrp` command can be used to switch their primary group.

AIX does not implement or use group passwords. It is not possible to log in using a group name.

3.1.1 Group usage for systems

A system may be a special case for group definitions. A system that is used only by a small number of users (or only one user) and that is never the target of `telnet` or `ftp` commands by anyone else is a special case. (It may also be a rare case because most workstations are members of a network and over time, for one reason or another, will be accessed by other users in the network.)

If meaningful group definitions will not be used for a workstation, then two of the AIX-defined groups should be used. These are the *system* and the *staff*

groups. Users (including yourself in your normal user mode) will be in the staff group. User root and yourself (in your administrator role) are in the system group.

3.1.2 User groups

In general, create as few user groups as possible.

Groups should be made for users who need to share files on the system, such as users who work in the same department, or users who are working on the same project.

You must create a group before you can assign users to it. Use SMIT to create new groups with the `smit groups` fast path.

Users can be a member of up to 32 groups; however, they can only have one primary group at a time. Users can change their primary group with the `newgrp` command. They must be defined as a member of a group before they can switch to it, of course. This may be important when creating files because, by default, the current group name is assigned to a newly created file. An alternate method of determining group ownership of a new file is discussed in 5.2.2, “Permissions bits - Advanced” on page 90.

The following are examples of the `newgrp` command:

- To change the real GID of the current shell session to payroll, enter:

  ```
  newgrp payroll
  ```

- To change the real GID back to your original login group, enter:

  ```
  newgrp
  ```

The `newgrp` command changes a user's real GID. When you run the command, the system places you in a new shell and changes the name of your real group to the group specified with the group parameter. By default, the `newgrp` command changes your real group to the group specified in the /etc/passwd file.

The user has access to files in all of the groups in their groupset. To list the groupset, use the `groups` or `setgroups` command.

Note

After issuing the command `newgrp payroll`, your primary group will not change to payroll. However, payroll will be reflected as the process group when executing the `setgroups` command.

Any user on the system can be defined as an admin user regardless of the group they are in. This sets the *admin=true* in /etc/security/user. Only root can remove a user who has the admin flag set.

There are two types of groups in the system: administrative groups and normal groups. An admin group is defined in /etc/security/group by the admin= true. In every group there can be a group-administrator.

For simple systems, it is recommended that you do not set the admin characteristic when creating groups. If a group has admin=true set in the /etc/security/group file, only the root user can administer that group.

The administrative parameters are confusing. If the value admin = true is in /etc/security/group, then this indicates an administrative group. But if a user has admin=true in /etc/security/user, this means that the user has administrative authority for that specific group, which is equal to the adms stanza in /etc/security/group. With admin = true, the user can administer that group.

The administrative group and authority has very little effect in AIX. We recommend you ignore the administrative groups and users. AIX has the *roles* functions to perform most things the root user can do without having root authority (refer to 3.2, “Administrative roles” on page 50). Or, for small systems, if your security policy permits it, another way to implement group administrators is to allow them to `su` to root.

3.1.3 System administration groups

There are two types of administrative groups, *system* and s*ecurity*:

system group
: System administrators should be members of the system group. System group membership allows an administrator to perform some system maintenance tasks without having to operate with root authority.

security group
: The security group is a system-defined group having limited privileges for performing security administration. The security group members have access to programs and files in /etc/security directory. Security group members can change most attributes for non-administrative users and groups, such as the user's login shell or the membership of a non-administrative group.

With little effort, a member of the security group can gain root authority; therefore, only trusted personnel should be in this group.

3.1.4 Changing default user groups

Your most common group name should be made the default group name for new users; as supplied by AIX, the default group name is *staff.*

To change the default group, edit the /usr/lib/security/mkuser.default file and change the *pgrp* stanza. This file provides default values for the `mkuser` command and SMIT. You cannot change the default group through SMIT.

For example, you might want your default group to be *office* instead of *staff*. Edit /usr/lib/security/mkuser.default and change:

```
user:
   pgrp = staff
```

To:

```
user:
   pgrp = office
```

New users added to the system (using SMIT) will default to group office. Of course, you may assign users to specific groups instead of taking the default.

3.1.5 System-defined groups

As distributed, AIX has groups that are needed by the system. Do not alter these groups unless you are very certain about what you are doing.

staff group
: This is the default group for all non-administrative users created in the system.

security group
: This is a system-defined group having limited privileges for performing security administration.

system group
: This is for selected system administrative tasks.

The other system-defined groups are used to control certain subsystems. Do not add users to these predefined groups unless you have a special purpose for doing so. The group IDs that come with the system are (in the form in which they appear in /etc/group):

```
system:!:0:root
staff:!:1:
bin:!:2:root,bin
sys:!:3:root,bin,sys
adm:!:4:bin,adm
```

```
uucp:!:5:uucp
mail:!:6:
security:!:7:root
cron:!:8:root
printq:!:9:
audit:!:10:root
ecs:!:28:
nobody:!:4294967294:nobody,lpd
usr:!:100:guest
perf:!:20:
shutdown:!:21:
imnadm:!:200:imnadm
```

We strongly advise you to not assign users to any existing group (except staff) unless you are certain of the consequences. Some of these groups (such as *bin* and *cron*) are the group owners of critical files and directories. Think twice before assigning a user to the system and security groups. A user in any of these groups, with a little effort, can subvert other security controls in the system.

The /etc/group file should grant read (r) access to all users and grant write (w) access only to the root user and members of the security group.

3.2 Administrative roles

Roles consist of authorizations that allow a user to execute functions that would normally require root user permission. These roles, rather like special groups, allow for non-root users to be assigned portions of root privileges. Roles virtually eliminate the need to log on as root since they provide for almost all the common administration functions. There are nine valid roles shown in Table 2.

Table 2. Roles and associated commands

Roles	Commands able to perform	Group
ManageBasicUsers	chsec, chuser, lsuser, mkuser	security
ManageAllUsers	chfn, chsec, chuser, mkuser, rmuser chrole, mkrole, lsrole, rmrole chsec, lssec, pwdadm chgroup, chgrpmem, chsec, mkgroup, rmgroup, chsec, chuser, lsuser, mkuser	security
ManageBasicPasswords	pwdadm	security
ManageAllPasswords	chsec, lssec, pwdadm	security

Roles	Commands able to perform	Group
ManageRoles	chrole, mkrole, lsrole, rmrole	security
ManageBackupRestore	backup, restore	No group
ManageBackup	backup	No group
ManageShutdown	shutdown	shutdown
RunDiagnostics	diag	system

Refer to 3.2.3, “Managing backup and restore roles” on page 53 for further information on ManageBackup roles.

Note

- The roles *ManageBasicPasswd* and *ManageAllPasswd* only support the `pwdadm` command. Therefore, you must know the old password for the user before you can change it. It does not support the `passwd` command.
- The roles ManageBasicUsers and ManageAllUsers cannot change passwords. Therefore, the root administrator will be required to set a new user’s password for the first time. The root user will also be required to reset a user’s password if the user has forgotten it.

Roles perform a lot of functions that can save the root administrator a lot of time; however, roles do not do everything.

Important

ManageBasicUsers, ManageAllUsers, ManageBasicPasswds, ManageAllPasswds, and ManageRoles require the user to be a member of the Security group with all of the power that comes with this group. We recommend only issuing these roles to users who are trusted and are security administrators.

3.2.1 Adding a role to a user

The `mkuser` command (when first creating a user) and the `chuser` command (when adding/removing a role to an existing user) are used to set up roles for users.

To add a role to a user, use the `smit mkuser` fast path:

```
* User NAME                                          [joe]
 User ID                                          []
```

```
ADMINISTRATIVE USER?                                  false
Primary GROUP                                        [security]
Group SET                                            [security,staff]
ADMINISTRATIVE GROUPS                                []
ROLES                                                [ManageAllUsers]
```

The `mkuser` command creates an entry in the /etc/security/user.roles file for each new user when the roles attribute is used. This file supports a default stanza. If an attribute is not defined, either the default stanza or the default value for the attribute is used.

An example of an /etc/security/user.roles file looks like:

```
default:
   roles   =
joe:
   roles = ManageAllUsers
jane:
   roles = ManageBackupRestore
```

The /etc/security/roles file contains the list of valid roles with a following stanza. Each stanza is identified by a role name followed by a colon (:) and contains attributes in the form Attribute=Value.

An example of a stanza in /etc/security/roles:

```
ManageBasicUsers:
   authorizations  =       UserAudit,ListAuditClasses
   rolelist        =
   groups          =       security
```

3.2.2 Adding/changing roles

It is possible to change or create new roles by using the following fast paths:

smit chrole To change the attribute values

smit lsrole To display the attributes and their values

smit mkrole To creates an entry for each new role in the /etc/security/roles

smit rmrole To remove a role

As distributed, the default roles provided in AIX allow almost all of the root functions required by a system administrator. We do not recommend creating new roles or changing or removing existing roles.

3.2.3 Managing backup and restore roles

Users in the ManageBackup and ManageBackupRestore roles can view and modify any file on the system. This includes the password and other security-oriented files. Be sure that trustworthy users are placed in these roles.

For some customer environments, it is required that the device used in backing up and restoring the entire system be protected from other users. The steps below help you make certain that you set up the system backup and restore correctly.

The following recommendation may prove helpful as you set up your system to perform backup and restore:

1. Create a group called *backup* using the `smit mkgroup` fast path.
2. Assign the ownership of the system backup and restore device to *root* user and group *backup* with mode 660 using the `chown` command to assign ownership and the `chmod` command to change permission.
3. Assign users in the ManageBackup and ManageBackupRestore role to group *backup* using the `smit chuser` fast path.

This configuration allows only the root user and members of group *backup* to access the system backup device.

3.2.4 Role authorizations

Role authorizations are authority attributes for a user. These authorizations allow a user to do certain tasks. For example, a user with the UserAdmin authorization can create an administrative user by running the `mkuser` command. A user without this authority cannot create an administrative user.

There are two types of authorizations:

Primary authorization
: Allows a user to execute a specific command. For example, RoleAdmin authorization is a primary authorization allowing a user administrator to execute the `chrole` command. Without this authorization, the command terminates without modifying the role definitions.

Authorization modifier
: Increases the capability of a user. For example, UserAdmin authorization is an authorization modifier that increases the capability of a user administrator belonging to the group security. Without this authorization, the `mkuser` command only creates

non-administrative users. With this authorization, the `mkuser` command also creates administrative users.

Authorization to commands are listed in Table 3.

Table 3. AIX role defaults

Authorizations	Commands
Backup	backup
GroupAdmin	chgroup, chgrpmem, chsec, mkgroup, rmgroup
ListAuditClasses	no command
PasswdAdmin	chsec, lssec, pwdadm
PasswdManage	pwdadm
UserAdmin	chfn, chsec, chuser, mkuser, rmuser
UserAudit	chsec, chuser, lsuser, mkuser
RoleAdmin	chrole, mkrole, lsrole, rmrole
Restore	restore

Authorizations are, by default, placed into the /etc/security/roles file for each role stanza. Do not touch or alter these entries. They are an integral part of the roles function and are what gives access to a user to perform certain commands.

3.3 Verifying the user environment

Security implementation requires both definition and maintenance. The security of a system is measured by how well the security state is maintained, as well as by how well it was originally defined.

Several “check” commands (`grpck`, `usrck`, `pwdck`, and `tcbck`) and “list” commands (`lsuser` and `lsgroup`) are available for use by root or anyone in the security group. These commands can help you maintain your security environment.

The `grpck`, `usrck`, and `pwdck` commands require a flag to indicate whether the system should try to fix erroneous attributes.

Flags are:

-n Reports errors but does not fix them.

-p Fixes errors but does not report them.

-t Reports errors and asks if they should be fixed.

-y Fixes errors and reports them.

It is wise to run the commands using the `-t` flag first, which gives you control and a choice to either leave or fix any errors.

The grpck command

The `grpck` command verifies that all users listed as group members are defined as users, that the GID is unique, and that the group name is correctly formed. Other minor checks are also done. The -t flag causes the command to report errors and ask you for permission to fix them:

```
grpck -t ALL
```

This checks the group environment, and, if you answer yes to a prompt, it will erase the user IDs that do not exist or where stanzas in /etc/security/user have conflicting data.

The usrck command

The `usrck` command verifies many parameters of a user ID definition. The `-t` flag causes the command to report errors and ask for permission to take a standard fix. In some cases, it will disable a user ID by adding an expired expiration date to the user definition. The user's data is not affected. The user can be enabled again by removing the expiration date (using SMIT or directly editing /etc/security/user).

Use this syntax to report problems and ask if they should be corrected:

```
usrck -t ALL
```

Never try to correct root using this command. If your having problems with *root*, read 2.3.4, “Repairing root user” on page 21.

The pwdck command

The `pwdck` command checks authentication stanzas in /etc/passwd and /etc/security/passwd. If anything is wrong, the standard fix is to remove the stanza or create a /etc/security/passwd stanza with an asterisk (*) in the password field.

This syntax will report problems and ask if they should be fixed:

```
pwdck -t ALL
```

The lsgroup and lsuser commands

The `lsgroup` and `lsuser` commands are used internally by SMIT, but you can

also use them directly. Direct use may be more convenient when you want to place their output in a file. The commands are:

```
lsgroup -f ALL >> /tmp/check
lsuser -f ALL >> /tmp/check
```

In the form shown here, these commands create the file /tmp/check and write their output into it. There is too much output for direct display on the screen, so the output would normally be directed to a file. These commands display most of the control information about users and groups. These commands may be used by any user, but much more information is displayed when they are used by root or any member of the security group.

The `lsuser` command is directly useful when used by root for a specific user:

```
lsuser joe
```

This command displays several lines containing control information for user *joe*. When used with the ALL operand, information is displayed for all users in the system. Several formatting options are available. You could write local programs to extract and display locally-important information obtained from these commands.

The tcbck command

The `tcbck` command is described in detail in Chapter 6, “Trusted computing base (TCB)” on page 109.

Chapter 4. Passwords

The security of any system is as good as its weakest password. Once a person has access to one account, it is possible to exploit weaknesses in the system configuration and gain access to root's account or mount a denial of service attack from the system.

When a new user is added with SMIT, AIX automatically disables the account by placing an asterisk in the second field (the password field) of the /etc/passwd line for the new user. The administrator (root) must use SMIT or the `passwd` command to set an initial password for the user. Because the password was set by root, the new user will be asked to change it the first time they log into the system. The *ADMCHG* flag in the user's entry in /etc/security/passwd indicates a password change is required. Setting the initial password enables the new account, permitting the new user to log into the system.

The `passwd` command is the "normal" UNIX command for changing passwords. The command can be used by any user to change their own password, or by root to change any user's password. There is no particular advantage to using SMIT to change a password; the `passwd` command does the same thing.

The SMIT function for changing a password is in the same menu as the function for adding a new user. It is usually convenient for the administrator to set an initial password for a new user immediately after they create the new user, and the SMIT function is convenient then.

In some cases, it may be appropriate to create a number of new users but not enable them (that is, not assign initial passwords). For example, a new group of student user IDs might be created at convenient times but not enabled until their class begins. In this case, using the `passwd` command may be more convenient than setting the passwords through SMIT.

4.1 Password issues

The administrator (running as root) must always assign an initial password in order to activate a new account. A new account cannot be used until this is done. There is a well-known exposure here. What password should the administrator set as the initial password? Administrators tend to set a common password, such as the user ID or a department name, for all new users. Knowing this, anyone can "steal" a new account by being the first to log

into the account (using the standard initial password). There are two solutions for this problem:

- The administrator can set an obscure password (different for every new user) and inform the user of the selected password.
- The administrator can delay setting the initial password until the new user is ready to log in. This means there will be a shorter period when a standard initial password is exposed.

In either case, the new user is prompted to alter their password as soon as they log into the system.

4.1.1 Password guidelines

Many security failures begin with poor passwords. Password quality has been discussed many times in many places; these discussions will not be repeated here. The following are basic guidelines:

- Do not use your user ID or any permutation of it.
- If you use the same password on more than one system, be extra careful with it. Never use the same root password on multiple systems.
- Do not use any person's name.
- Do not use words that can be found in the online spelling-check dictionary, especially for a networked or larger multi-user system.
- Do not use passwords shorter than five or six characters.
- Do not use swear words or obscene words; these are among the first words tried when guessing passwords.
- Do use passwords that you can remember. Do not write down your password.
- Do consider passwords that consist of letters and numbers.
- Do use passwords that you can type quickly.
- Two words, with a number in between, make a good password.
- A word (with at least six characters), with a numeric digit inserted in the word, is an excellent password. But do not form the digit by changing an "l" to "1" or an "o" to "0." A word with an internal digit is a better password than a word with a leading or trailing digit.
- A pronounceable password is easier to remember.
- AIX checks only the first eight characters of the password; however the word can be longer than eight characters.

- A good scheme is to memorize a sentence and use the first letter of each word for the password. For example, “the cat sat on the mat” = tcsotm.

Restrictions should be set so that passwords are hard to guess, yet not hard to remember. Passwords that are hard to remember are often written down somewhere, which compromises system security.

4.2 Password defaults

Each user account has a set of associated attributes. These attributes are created from default values when a user is created.

The default values provide no password quality controls. This is intentional, since it conforms with the expected characteristic of standard UNIX.

You can specify password quality and composition rules for all users by editing the default stanza in /etc/security/user. Individual controls can be set using the SMIT menu discussed in 4.2.1, “Enforcing strong passwords” on page 60.

Recommended, default, and maximum password attribute values listed in Table 4.

Table 4. Password attribute values

Restriction Values	Recommended Values	Default Values	Maximum Values
minage	0	0	52
maxage	12 (5 weeks for root user)	0	52
maxexpired	4	-1	52
minalpha	4	0	8
minother	1	0	8
minlen	6 (8 for root user)	0	8
mindiff	3	0	8
maxrepeats	3	8	8
histexpire	26	0	260*
histsize	8	0	50
pwdwarntime	14	0	0

Restriction Values	Recommended Values	Default Values	Maximum Values
*A maximum of 50 passwords are retained. -1 means the feature is disabled.			

4.2.1 Enforcing strong passwords

The most common method of an intruder gaining access to a system is by password guessing, and many users help in this by choosing weak passwords.

By using the facilities provided by AIX, it is possible to force users to choose passwords that are hard to guess even by using a dictionary.

A way to do this is to set up the password restrictions as system defaults before any user accounts are added. This can be achieved by editing the /etc/security/user file. Or, you can set up enforced passwords for individual users using the `smit mkuser` and `smit chuser` fast paths. Refer to Table 4 on page 59 for suggestions on recommended values.

minage

Minimum number of weeks that must pass before a password can be changed. We recommend that you do not use the *minage* parameter. It can create awkward situations and may cause more trouble than it cures.

maxage

Maximum number of weeks that can pass before a password must be changed. Consider using smaller *maxage* values for privileged users, such as root and members of the system group. The maxage of a password limits the time period during which an exposed (or disclosed) password can be used.

There has been some debate whether a rigid password expiration period is a good option. If a user suddenly must select a new password, the user may select a trivial or poor password. That is, the user may not be prepared to seriously think about a new password while he or she is trying to log into the system for some other purpose. The *pwdwarntime* parameter (specified in days) causes AIX to warn the user shortly before the password expires. This permits the user to change his or her password in a timely manner.

maxexpired

The maximum number of weeks beyond maxage that a password

can be changed before administrative action is required to change the password. Root is exempt.

minalpha
Minimum number of alphabetic characters the new password must contain.

minother
Minimum number of non-alphabetic characters the new password must contain. Other characters are any ASCII printable characters that are non-alphabetic and are not national language code points.

minlen
Minimum number of characters the new password must contain.

Note

The minimum length of a password on the system is *minlen* or *minalpha* plus *minother*, whichever is greater. The maximum length of a password is eight characters. *minalpha* plus *minother* should never be greater than eight. If *minalpha* plus *minother* is greater than eight, then *minother* is reduced to eight minus *minalpha*.

maxrepeats
Maximum number of times a character can appear in the new password.

mindiff
Minimum number of characters in the new password that must be different from the characters in the old password.

histexpire
Number of weeks that a user will not be able to reuse a password.

histsize
Number of previous passwords that cannot be reused.

Note

If both *histexpire* and *histsize* are set, the system retains the number of passwords required to satisfy both conditions up to the system limit of 50 passwords per user. Null passwords are not retained.

pwdwarntime
Specified in days, causes AIX to warn the user shortly before their password expires.

account_locked

The number of consecutive unsuccessful login attempts the user is allowed. If this number is exceeded, the account is locked and the user cannot login. If 0 is specified this feature is disabled.

Note

To unlock a user's account that was locked because of too many failed logins, the system administrator can use the Reset User's Failed Login Count menu item by using the `smit users` fast path.

loginretries

The number of invalid login attempts before a user is not allowed to log in. Possible values: a positive integer or 0 to disable this feature.

dictionlist

List of dictionary files checked when a password is changed. Dictionary files contain passwords that are not allowable. The AIX 4.3 dictionary list is located in /usr/share/dict/words. The root user can add or delete words from this file by editing it. Users will not be able to select a password that is contained in this file. Only available if text processing is installed, fileset bos.txt.spell. This is a good security tool and we highly recommend using it.

pwdchecks

List of external password restriction methods that are used when a password is changed. Refer 4.4, “Extending password restrictions” on page 65.

4.3 Password variations

Although, in theory, it is possible for an intruder to keep entering passwords until the right one is guessed, this would be very time consuming when done manually. However, when automated, based on dictionary attack programs, a password guessing program will often succeed in finding passwords for several users in any larger UNIX systems.

This can be prevented in a variety of different ways as per the following sections.

4.3.1 Dictionary attacks (crackers)

Dictionary attacks can only be performed by a privileged insider since the user must have read access to the /etc/security/passwd file. These programs work by guessing the password for a user's account. The program reads encrypted passwords from /etc/passwd files and attempts to guess passwords that, when encrypted, will match a password in the file. Typically, the intruder provides a list of trial words (a "dictionary"), and the cracker program will try each word (and a number of permutations of each word) in the list. These programs are known as "dictionary attacks".

In normal use, AIX does not maintain encrypted passwords in /etc/passwd; rather, they are maintained in /etc/security/passwd in a different line format. Nevertheless, most cracker programs can be readily modified (if the source code is available) to work with /etc/security/passwd. The user must have read access to this file, of course.

If you (an administrator) have a cracker program available and have some time to work with it, we recommend using it. It cannot harm your system (assuming you have a trustworthy program) and may find any number of poor passwords set by your users.

One useful program for this is called "john the ripper". This is available at:

```
http://www.bull.de/pub/out/
```

To run john the ripper on your system, you must have access to the /etc/security/passwd file (however, we do not recommend granting read access to /etc/security/passwd to any person who is not an authorized administrator). John the ripper works by merging the contents of /etc/security/passwd with /etc/passwd by putting the encrypted password in the second field of /etc/passwd. John the ripper then submits a list of names from dictionaries to a variation of the crypt(3) algorithm. It then compares the result to the contents of the modified /etc/passwd. If a system administrator performs this on their system, the weak passwords along with the UID are found very quickly.

Note

Activating the /usr/dict/share/words file by adding this path to the *dictionlist* variable in /etc/security/user file will help prevent "weak" passwords. Refer to 4.2.1, "Enforcing strong passwords" on page 60.

4.3.2 One-time passwords

Dictionary attack programs and intruders rely on the premise that the password obtained can be reused later. A popular solution to this problem is to use a one-time password system in which the password changes in an unpredictable manner every time it is used.

There are many one-time password programs available, (not directly from IBM). One of the most popular is offered by SecurID. The users carry a small device not much larger then a business card with an LED display. The display changes every 60 to 90 seconds. When the user requires to login they enter the currently displayed password. The mobile unit is synchronized with the server and will thus know the password.

4.3.3 Two-person login

One common method of increasing login security is to require two passwords to authenticate an account. This is called “2 key authentication” and can be enabled through SMIT. The assumption is that two different people (with the two different passwords) must be present to complete the login. The two different passwords are associated with two different accounts. There is no way, using the standard facilities, to maintain two passwords with a single account. For example, change the PRIMARY authentication method field of user joe by `smit chuser` fast path as follows:

```
PRIMARY authentication method                    [SYSTEM,SYSTEM;jane]
```

When *joe* logs into the system, in this example, he will be prompted for his password. If he responds correctly, the system will issue a prompt for *jane's* password.

The prompts would be:

```
login: joe
joe's password: xxxxxxxx
jane's password: xxxxxxxx
```

You must set the PRIMARY authentication method exactly as shown in the preceding SMIT screen. The SYSTEM parameter specifies that the normal password authentication program should be used. By default, it checks the password of the user currently logging into the system. The second SYSTEM parameter specifies a second check. In this case, it has an operand, *jane*, and verifies the password for the account specified in the operand. The syntax is unusual: a comma separates two different authentication steps. A semicolon separates an authentication method (SYSTEM) from an optional account name.

4.4 Extending password restrictions

The operating system provides a method for administrators to extend the password restrictions. Using the *pwdchecks* attribute of the /etc/security/user file, an administrator can add new subroutines (known as methods) to the password restrictions code. Then, local site policies can be added to and enforced by the operating system.

The *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 1*, SC23-4159 contains a description of the pwdrestrict_method, and the subroutine interface that specified password restriction methods must conform to. To properly extend the password composition restrictions, the system administrator must program this interface when writing a password restriction method. Caution is advised in extending the password composition restrictions. These extensions directly affect the `login` command, the `passwd` command, the `su` command, and other programs. The security of the system could easily be subverted by malicious or defective code. Only use code that you trust.

4.4.1 Additional authentication - auth1 and auth2

AIX allows you to specify additional primary authentication steps (methods) and secondary authentication steps. In AIX terminology, a primary authentication method can reject a user login; a secondary authentication method cannot reject a login. A secondary authentication step is a method of running a specific program (which may have nothing to do with authentication) as part of a specific user's login process. (This terminology is unique to AIX). Commands `login`, `telnet`, `rlogin`, and `su` support these authentication methods.

Password and local program

You can add a local program that provides additional authentication. Your program obtains control during a login process. If your program ends with return code zero, the login process continues. If your program ends with any other return code, the login process is terminated with the message "`You entered an invalid login name or password`." The additional authentication program runs as root; it need not have SUID.

Figure 6 on page 66 shows an example using a simple program.

Note

The authentication method defined here is provided as an example only. The program we have selected is a simple challenge/answer program. It is written in C, but another language, or shell script, can be used. When the user attempts to log in, they will be presented with a number. The user must manually calculate the last two digits of the number multiplied by 17, and type in the answer. If the answer is correct the login continues, if incorrect the login will fail.

Steps to set up Primary Authentication:

1. Using an editor, create a program or script such as the one shown in Figure 6.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

int main() {
    long challenge, asked, answered;
    char xxx[16];

    srand48 (time(NULL));
    challenge = lrand48();
    printf("%ld\n", challenge);
    asked = challenge % 100 * 17;
    scanf ("%4s", xxx);
    answered = atol(xxx);
    if(asked == answered)
        return 0;
    else
        return -1;
}
```

Figure 6. Example for the additional authentication program

   This program (after you compile it and place the output in a convenient location—we used /usr/local/bin/challenge) must be defined as an authentication method.

2. Save the compiled program and then change its permissions. Make sure this file is owned by root and does not have write or read permission for group or other.

   ```
   chmod 711 /usr/local/bin/challenge
   ```

3. Edit /etc/security/login.cfg to add the following lines:

   ```
   auth_method:
   ```

```
program = /usr/local/bin/challenge
```

The authentication method name (/usr/local/bin/challenge) need not be the same as the program name (/usr/local/bin/challenge) as in this example).

4. Edit the /etc/security/user file or use SMIT to add this authentication to one or more users.

   For example, auth1 has been added to user joe's stanza:

   ```
   joe:
      admin = false
      auth1 = SYSTEM,auth_method
      auth2 = NONE
   ```

   Alternatively, change the PRIMARY authentication method field of user joe by the `smit chuser` fast path as follows:

   ```
   PRIMARY authentication method [SYSTEM,auth_method]
   ```

   This change will affect joe only.

   If adding it to the defaults stanza of /etc/security/users, it should be well tested before doing this. For example:

   ```
   root: auth1 = SYSTEM
   user1: auth1 = SYSTEM,auth_method
   user2: auth1 = SYSTEM,auth_method
   ```

Note

Root should be an exception to this default.

When user joe next logs into the system, he should receive the following prompts:

```
Console login: joe
joe's Password         Joe enters his password as usual
1858929184             A number like this will appear
1428                   Joe's input and enter (84*17). Login complete.
```

Joe will have to multiply the last two digits by 17. If the answer is correct, the login will succeed. Any other response will cause the login to be rejected with an error.

If you specify auth_method only as the authentication method (that is, remove the SYSTEM parameter), the login process will first prompt for the calculation. It will then prompt for the standard password. That is, the SYSTEM authentication (using the standard password) apparently is always used, whether or not it is specified.

Do not delete the *auth1* parameter from the default stanza in /etc/security/user. If you do, you will be unable to log into any account that does not have an additional authorization method defined.

If you experiment much with various authentication methods and parameters, you will probably lock yourself out of your system. If this happens, review 2.3.4, “Repairing root user” on page 21. A good suggestion is to test the root recovery process before you start working with authentication parameters.

We recommend using additional authentication only if there is a specific need for it.

4.5 IBM RS/6000 hardware passwords

IBM RS/6000 provides the following three hardware (including firmware) security features:

Cover lock key

This security feature prevents the cover from being removed. You need a physical key to access the inside hardware components.

Power-on password

This password helps protect information stored in your system. Every time you power on or reset your system, this password is required to continue the operation.

When the system is powered on, it checks whether a power-on password (POP) is present. If there is one present, and the “unattended start mode” is not set, it means the machine's owner does not want the system to be used unless the POP is supplied. In this case, the system will prompt for the POP. The user is given three attempts to enter the correct password. If the user fails to supply the correct password, the system will go into a “hung” state and must be powered off before continuing. This password helps protect information stored in your system.

Unattended start mode

To use this mode, a power-on password must be previously specified. If unattended start mode is enabled, the system will boot from the defined boot device without requiring the user to enter the power-on password. While the system can be booted without entering the POP, the keyboard controller will be programmed to lock up until the POP is provided. This mode is ideal for servers that run unattended. After an electrical power failure, for example,

the operating system will be rebooted without waiting for a user to enter the power-on password.

Privileged-access password

This password protects against the unauthorized starting of System Management Services (SMS). SMS is built-in firmware that provides system management tools that include setting or resetting power-on/privileged-access passwords.

When the user presses one of the keys to access SMS, the system will check to see if a privileged access password exists; if it does, the user is prompted to enter the privileged access password. The user is given three attempts to supply the correct password. If the user fails to do so, the system will go into a “hung” state and must be powered off before continuing.

If you set both power-on and privileged-access passwords, only privileged-access password is required to start SMS. Password setting and the required password to start AIX or SMS are summarized in Table 5.

Table 5. Password setting and required passwords

Password setting	Starting AIX	Starting SMS
None	Not required	Not required
Power-on	Power-on	Power-on
Privileged-access	Not required	Privileged-access
Both power-on and privileged-access	Power-on	Privileged-access

In case you don't have a machine's password, the only way to get access to the system is by removing the system's battery. You must be aware that this procedure will erase all firmware configuration data maintained in NVRAM, such as the error log and any configured IP addresses. In this case, you need the cover lock key to open the cover.

Note

Power-on passwords only apply to PCI-based RS6000 machines. The implementation of these hardware security features are slightly different between RS/6000 models. For more precise information, refer to the User’s Guide distributed with your RS/6000 system.

We recommend that both power-on and privileged-access passwords are set, and the cover lock key is removed from RS/6000. The cover lock key must be available when it is needed for software or hardware maintenance.

Depending on your system application, you may not need to use the power-on password. Even in such a case, a privileged-access password should be set. Nevertheless, anyone can start SMS and bypass all security and access any file on the disks.

If you decide not to use a power-on password, we recommend you change the boot device sequence. As distributed, RS/6000 searches for operating system start up code in the following sequence (if available):

1. Diskette drive
2. CD-ROM drive
3. Hard disk drive
4. Network device

This means anyone can boot your RS/6000 from their own start up code provided by a diskette or CD-ROM. Such a code could bypass all security and access any file on the disks. Actually, if you forget root's password, you may need to use this procedure. For more information, refer to 2.3.4, "Repairing root user" on page 21. Therefore, we recommend you specify only the hard disk drive as a boot device. Setting it this way allows RS/6000 boots from only AIX on the hard disk.

4.6 Automatic logins

On some systems, it is an application that presents the front end to the user. The user is prompted to identify themselves to the applications, then the application logs the user on. The application may contain the user ID and password in plain text form. If this is the case, the application files must be treated as if they were the /etc/security/passwd file itself.

Section1.2.4, "Application security" on page 12 touches on considerations for application security; however, for detailed information, refer to your application vendor.

Chapter 5. File, directory, and file system security

Computer security is very similar to other types of security. Its goal is the protection of information stored on the computer system, a valuable resource. Information security is aimed at:

Integrity The value of all information depends on its accuracy. If unauthorized changes are made to data, this data loses some or all of its value.

Privacy The value of much information depends on its secrecy.

Availability Information must be readily available.

Other than the login process, file security is the most apparent element of security to most AIX users. The discussion in this chapter considers only local files, files that are not accessed through a LAN or remote connection. The basic elements controlling file security are:

- The permission bits associated with the file
- The permission bits associated with the directory containing the file name
- The permission bits in all the directories in the file's path
- Extended access control list parameters, if any
- The owner of the file
- The group-owner of the file
- The owner and group-owner of the file's directory
- The owners and group-owners of all higher-level directories in the file's path
- Programs executing with the effective user ID of root

The individual elements are not complex, but the effect of various combinations can be confusing. This chapter discusses each of the listed elements.

5.1 File systems

UNIX documentation discusses "files" and "file systems." This chapter is about files, but it is important to understand the terminology involved.

There are two common uses for the term "file system" in AIX. One is the total view or complete tree structure incorporating all the files from the top or root (/) directory, including all the other directories and files. This usage of the

term is technically incorrect, but is widely used. For example, you may hear a user say something like: “I do not want any TCP/IP user to access my file system.” “My file system never seems to have `fsck` problems.” These two quotes informally refer to all the files on the owner’s systems. They are using “file system” to mean “all my files.”

The technically correct meaning of “file system” in AIX is a contiguous space on a disk (or partition of a disk), or a logically contiguous disk area managed by a Logical Volume Manager (LVM) that contains all the controls and control blocks needed to manage its own internal space, including the allocation of files and directories. In the case of AIX, one can equate “file system” with “logical volume.”

AIX recognizes several types of file systems and can have multiple instances of all of these in a given system:

- Journaled File System (JFS), the normal AIX file system.
- Network File System (NFS), for file sharing across networks.
- CD-ROM File System, for reading CD-ROM disks.
- Distributed File System (DFS), an optional component of the Distributed Computing Environment (DCE).
- Andrews File System (AFS), an optional file system that supports a number of UNIX systems.
- Raw disk volume, not containing a standard file system. A number of database and other types of products use raw disks. The system may use raw disk volumes for paging space, dump space, and other specialized uses.
- Diskettes do not normally contain file systems.

An NFS and a CD-ROM file system are special cases and are not discussed here. The JFS is the standard AIX file system. The “journal” part of the name can be misleading since “journal” is often a database-related term. AIX does “journal” (in the proper database sense) the changes to inodes. An inode is a control block in a file system that keeps track of various pointers to files and free space. That is, inode changes are protected from corruption due to system failures. This is a substantial improvement over older UNIX systems that were very sensitive to file system (inode) damage. The journal action does not apply to data in files. Data integrity in AIX is similar to that in other UNIX systems. Of course, database products running under AIX may perform their own journal functions for their data.

Generic UNIX documentation often refers to a local file system. For AIX, this is a JFS. There are also references to a virtual file system (VFS). This is simply a coding and design technique that provides a uniform programming interface regardless of what type of actual file system is being used. JFS, DFS, NFS, and CD-ROM file systems are accessed through the VFS interface, providing a common API for the user. From the user and administrator point of view, VFS is all under-the-covers and requires no action.

The “journal” part of JFS works automatically. There is no user or administrator involvement. The `fsck` command has been modified to invoke JFS recovery checking. Booting AIX automatically runs `fsck` (including the JFS recovery functions), and no other actions are normally required. Refer to 3.3, “Verifying the user environment” on page 54 for more information on the `fsck` command.

A basic AIX system has several logical volumes as follows:

```
$ lsvg -l rootvg
rootvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd5                 boot       2     2     1    closed/syncd  N/A
hd6                 paging     32    32    1    open/syncd    N/A
hd8                 jfslog     1     1     1    open/syncd    N/A
hd4                 jfs        3     3     1    open/syncd    /
hd2                 jfs        201   201   1    open/syncd    /usr
hd9var              jfs        1     1     1    open/syncd    /var
hd3                 jfs        3     3     1    open/syncd    /tmp
hd1                 jfs        1     1     1    open/syncd    /home
```

Notice that a separate logical volume (hd8 in this example) is used by JFS as the journal area. The special logical volumes and the logical volume manager itself are normally manipulated through SMIT, and this requires root authority. Other than protecting the root password, there is no routine security administrative work involved with the special logical volumes.

5.1.1 Mounting files and directories

A file system must be mounted before it can be used. File systems can be mounted automatically on startup as specified in the /etc/filesystems file or as needed by using the `mount` and `umount` commands. Only users with root authority or who are members of the *system* group can mount or unmount file systems. Either the `smit mount` fast path or the `mount`/`umount` commands may be used.

The /etc/filesystems file

The /etc/filesystems file should include a stanza for each mountable file system, directory, or file. This stanza should specify at least the name of the file system and either the device on which it resides or the directory name. If the stanza includes a mount attribute, the `mount` command uses the associated values. It recognizes five values for the mount attributes: automatic, true, false, removable, and read-only.

The `mount all` command causes all file systems with the mount=true attribute to be mounted in their normal places. This command is typically used during system initialization, and the corresponding mounts are referred to as automatic mounts.

An example of a typical stanza in /etc/filesystems file is shown as follows:

```
$ pg /etc/filesystems
/home:
        dev             = /dev/hd1
        vfs             = jfs
        log             = /dev/hd8
        mount           = true
        check           = true
        vol             = /home
        free            = false
```

Attention

Modifying this file can affect the file systems in several ways.

The mount command

The `mount` command instructs the operating system to make a file system available for use at a specified location (the mount point). In addition, you can use the `mount` command to build other file trees made up of directory and file mounts.

Any user can execute the `mount` command with no arguments. This will return the list of currently mounted file systems (JFS, NFS, and DFS).

The `mount` command, entered without flags, displays the following information for the mounted file systems as shown:

```
$ mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    May 02 17:23 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    May 02 17:23 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    May 02 17:23 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    May 02 17:23 rw,log=/dev/hd8
```

```
/dev/hd1        /home           jfs    May 02 17:24 rw,log=/dev/hd8
```

- The node (if the mount is remote)
- The object mounted
- The mount point
- The virtual-file-system type
- The time mounted
- Any mount options

Only users with root authority or who are members of the *system* group and have write access to the mount point can issue file or directory mounts. The file or directory may be a symbolic link. The `mount` command uses the real user ID, not the effective user ID, to determine if the user has appropriate access. System group members can issue device mounts, provided they have write access to the mount point and those mounts specified in the /etc/filesystems file. The file system must be currently unmounted.

Users with root authority can do almost anything with `mount`, mounting any file system over any directory and overriding any parameters in the /etc/filesystems.

Non-root users or users who do not have system group authority are unable to mount anything at all. For example, they cannot mount directories over other directories to which they have write access. This function was available in earlier releases of AIX but is no longer available in AIX v4.3.

CD-ROM file systems

CD-ROMs present, potentially, a serious security exposure. Files can be executed from a CD-ROM, including SUID root files. Devices for writing CD-ROMs are commonly available. Someone could create a dangerous SUID root program on an external system and create a CD-ROM containing this file. If mounted on your system, the dangerous program will execute with SUID root authority. You cannot control the external creation of CD-ROMs. However, you can control how CD-ROMs are mounted on your system. When you mount a CD-ROM, you can specify `mount -o nosuid`. You should use this when appropriate; you should probably include this option in any batch files you create to mount CD-ROMs. It is also advisable to put your machine in a limited access area where users cannot access the CD-ROM device. Alternatively, you can mount a file system on a CD-ROM. This will not allow any user to eject the CD-ROM media and insert their rouge media.

5.1.2 Private file systems

Adding a private file system involves creating a new logical volume also. This is usually done through SMIT.

Simple systems, using basic AIX facilities, may not need any private file systems. A user can store their private data in their home directory (which is normally /home/username) and in additional directories created below the home directories. We recommend that you discourage the use of private file systems on workstations unless there is a particular need for them.

Major software products, such as a database package or an office systems package, often reside in private file systems. If such products are installed with the system, private file systems may be required. The install process for these products may create the file system.

Servers, typically with several application products installed (such as database managers), often have many file systems in addition to the basic AIX file systems. The administrator (working as root) must create whatever additional file systems are required. This may require a certain amount of disk space planning. Some understanding of LVM is required. Disk space planning is not discussed here. In general, the administrator will select (through SMIT) to have the additional file systems automatically mounted whenever the system is booted.

The administrator must be concerned with the security controls of the new file systems. The ownership and permissions of the mount point directory are important since all the contents of the file system will be under this directory. The file security management discussions in this chapter apply to locally-created file systems, as well as to system-created file systems.

If you create additional file systems, we recommend that the directory mount points have permission bits of -rwx------- (octal 700). Refer to 5.1.7, “Changing file permissions” on page 84 for details on changing permissions. Understand that you mount “over” the mount point directory, so any user files or directories below the mount directory are over laid by the newly mounted file system and will not be available until you unmount the file system again. Installations with extreme security requirements can use a portable file system that is unmounted and disconnected from the computer when not needed.

Removable file systems (which are usually “private”) have a unique exposure. When mounted, they function as normal file systems, including *SUID* and especially *SUID root* functions. A user could take a portable file system somewhere else and add many SUID root programs. When mounted on your

system, all these SUID root programs could be disastrous. You have some control over mounting private file systems, since root, or system in some cases, authority is needed to mount. One of the mount options is `-o nosuid`. We recommend you always use this option when mounting portable file systems (including CD-ROMs) and (possibly) when mounting any private file system. You should also include this option in any batch files you create to mount CD-ROMs.

AIX does not support JFS (or any other easily used file system) on diskettes. AIX can read and write DOS-formatted diskettes with the `dosread` and `doswrite` commands (bos.dosutil fileset). Diskettes in tar format can also be used.

5.1.3 Inodes and links

The normal UNIX file systems (including AIX JFS) use a level of indirect file control that is usually hidden from the user. The administrator must understand some of the basic elements of this, since they are important for file security.

UNIX file access is usually like this sequence:

```
directory entry -> inode -> data blocks
```

That is, the directory entry for a file does not point to the data for the file. It points to an inode that, in turn, points to the data. This discussion ignores the details of inode data and indirect addressing through inodes. These details are not relevant to routine security processes.

The security permission bits are attached to the inode, not the directory entry. Also, multiple directory entries may point to the same inode. A directory entry contains a name for a file, such as /home/trial/data. An inode has an identification number, but no file name. (A number of low-level UNIX commands exist to manipulate inodes directly. These commands should not be used by non-administrator users, although they do not bypass any security functions.)

A more general picture might be:

```
/home/trial/data  -->
/xyz/j/g34/check  --> inode 317 --> data blocks
/joes/stuff       -->
```

In this example, a single file (based on inode 317 within some file system) has three directory “links.” The same file has three very different “names.” Permission bits (and the UID and GID) are stored in the inode. Accessing the

file through any of the names will provide the same permissions and owner controls. These extra names are provided by symbolic links or hard links:

Symbolic link A symbolic link can function across file systems and is not deleted if the target inode is deleted.

Hard link A hard link works only within a given file system and can be a controlling element in deleting the inode and file data.

Similar links can exist within directory levels. For example, the /xxx directory could be linked to the /etc directory. This means that file /xxx/my/data is really /etc/my/data.

AIX and other UNIX systems use symbolic links heavily. The base AIX system does some of this by default. The `ls -l` command denotes these with an arrow in the name field and an “l” (for link) as the first character of the permissions field. For example, there is no /u file system. Instead /u is linked to /home, for example:

```
$ ls -al /u
lrwxrwxrwx   1 bin      bin 5 May 01 11:13 /u -> /home
```

The /home/her/data file can also be accessed as /u/her/data. The same file is accessed in both cases, although different directory structures are used.

The /u is a symbolic link to /home. Note that it appears that everyone has write permission to /u. This is misleading. The permissions in a symbolic link have no meaning. The effective permissions are taken from the target name. In the above example, anyone working with /u must work under the permissions set by /home. In this example, /u and /home are directory levels, but the same concept applies to both directories and files.

UNIX (including AIX) has no simple way to detect when the target of a symbolic link has been deleted. Over time, symbolic links with missing targets may accumulate. These can cause errors that puzzle normal users. No direct security concerns are caused by this, but you should be aware of the problem. The most common effect is that a file appears to exist when accessed by one method, but appears to be missing when accessed a different way.

5.1.4 Ownership

Initially, a file owner is identified by the user ID of the person who created the file. The owner of a file determines who may read, write (modify), or execute the file. Ownership can be changed with the `chmod` command. Refer to 5.1.6, “File and directory permissions - Basic” on page 81 for information on `chmod`.

Every user ID is assigned to a group and each group has a unique group ID. The group is the current group of the owner when a user created the file. When a new file is created, the operating system assigns permissions to the user ID that created it, to the group ID containing the file owner, and to a group called others, consisting of all other users. The `id` command shows your user ID (UID), group ID (GID), and the names of all groups you belong to. For example:

```
$id joe
uid=203(joe) gid=1(staff)
```

Only the root user can change the owner of a file by using the `chown` command and change the group owner with the `chgrp` command. You can change the group of a file only if you are a root user or if you own the file. If you own the file, but are not a root user, you can change the group only to a group of which you are a member.

In file listings (such as the listings shown by the `ls` command), the three groups of users are always represented in the following order: user, group, and others. If you need to find out your group, the `groups` command shows all groups for a user ID. For example:

```
$ groups joe
security staff
```

The group or group-owner is the GID of a group defined in /etc/group. Any member of this group (as defined in /etc/group) has whatever rights a group-owner has for the file.

To display group information by listing the attributes of all the groups on the system, or for a particular group, use the `lsgroup` command as shown:

```
$ lsgroup ALL
system id=0 users=joe,katie,root
staff id=1
users=joe,jane,leo,bruce,katie,ailsa,brian,julie,aaron,jenny,daemon
bin id=2 users=root,bin
sys id=3 users=root,bin,sys
adm id=4 users=bin,adm
uucp id=5 users=uucp
mail id=6 users=
security id=7 users=leo,reggie,root
cron id=8 users=root
printq id=9 users=
audit id=10 users=root
ecs id=28 users=
nobody id=-2 users=nobody,lpd
```

```
usr id=100 users=guest
perf id=20 users=
shutdown id=21 users=
imnadm id=200 users=imnadm
```

Copying and moving files

It is important to understand how the ownership of a file is affected by the `mv` and `cp` commands.

The `cp` command always creates a new file, and the user of the command becomes the owner of the new file. The user must have sufficient permissions to read the source files. This requires at least execute permission for all the directories in the path of the input file, and read permission for the file itself. The user must have write permission for the target directory.

If the `mv` command is used to move a file within a file system, the ownership of the file is not changed. The user of the `mv` command must have write permission in the target directory and sufficient permissions to read the file.

If the `mv` command is used to move a file to another file system, a new file is created in that file system, and the current user is the owner of the file. The user must have write permission in both the source directory to delete the file and the target directory to create a new file. The user must also have appropriate read permissions for the source.

5.1.5 Unowned files

"Unowned" files typically occur when users are removed from the system. When a user is removed, all of their files and the home directories remain. The user may also have files in other areas, for example, spool files and mailbox files.

These files, when listed using the `ls` command, contain a user ID rather than a user name. For example, if user joe is removed from the system, the `ls -al` command would be shown as follows:

```
# cd /home/joe
# ls -al
total 3
drwxr-xr-x   2 212      staff        512 May 05 12:36 .
drwxrwxrwx  19 bin      bin          512 May 04 09:15 ..
-rwxr-----   1 212      staff        254 May 01 11:38 .profile
-rw-r--r--   1 212      staff          0 May 05 12:36 bond007
-rw-r--r--   1 212      staff          0 May 05 12:36 project1
-rw-r--r--   1 212      security       0 May 05 12:36 topsecret
-rw-r--r--   1 212      staff          0 May 05 12:36 wishlist
```

The third field now contains a user ID rather than the user name joe.

There is no longer a user name associated with the UID, and the files are said to be “unowned.” If you add a new user with the same UID as the deleted user, the new user immediately becomes the owner of the files. This UID is the entry in the /etc/passwd file. A UID will not be reused by SMIT. This will only occur if you force it by editing /etc/passwd. This can cause security exposures.

The `find` command can be used to list all files owned by a specific user (before the user is deleted). For example:

```
find / -user joe
```

We recommend running this command before removing a user from the system. These files can then be checked, and useful ones allocated to other uses using the `chown` command. The remainder can then be deleted.

To check a system for unowned files:

```
find / -nouser
```

The listed files can be checked and reallocated (`chown`) or deleted as required. Be careful though since system files will also be included in this list, notably /dev/console. Do not delete these!

If NFS is used for remote file system mounting, but Network Information Services (NIS) is not used, difficulties can arise in identifying file owners. This is because files that belong to a user on system A, for example, when mounted on system B and viewed by a user on that system, appear to be unowned unless the user on system A is also know to system B. The owner may be incorrectly identified in some cases if conflicting user numbers exist on separate systems. Again, a file is considered unowned if the system (using an `ls` command, for example) displays a user ID of a user name for the owner.

To prevent the `find` command from searching file systems mounted through NFS, the option `-fstype jfs` can be added. Therefore, the `find` commands would become respectively:

```
find / -user username -fstype jfs
find / -nouser -fstype jfs
```

5.1.6 File and directory permissions - Basic

Files (and directories) have permission bits. An administrator must thoroughly understand these. These are sometimes called *mode bits*, but we will use the term *permission bits* or *permissions*.

There are three classes of users: user/owner, group, and all others. Access is granted to these groups in some combination of three modes: read, write, or execute permission. The basic permission bits are quite simple.

There are 12 permission bits:

- Three system bits (5.2.2, “Permissions bits - Advanced” on page 90).
- Three *user*/*owner* bits that describe what the owner is permitted to do.
- Three *group* bits that describe what any other user who is a member of the group may do.
- Three *other* or world bits that describe what any other user can do.

Permission bits are often displayed as nine bits. (The three high-order system bits are displayed in special ways.) A typical permissions display is:

```
rwxr-xr--
```

The first three characters displayed are the *user* bits, the next three are the *group* bits, and the last three are the *other* bits.

By convention, a letter means the bit is on, and a dash means it is off. In the example above, the owner has read, write, and execute permission, anyone in the file’s group has read and execute permission, and everyone else has read permission. Permissions are not hierarchical; write does not include read, and execute does not include read.

Table 6 illustrates the default file access modes for the three sets of user groups.

Table 6. Default file permissions

File classes	Read	Write	Execute
u = User/Owner	Yes	Yes	Yes
g = Group	Yes	No	No
o = Others	Yes	No	No

Table 7 illustrates the default directory access modes.

Table 7. Default directory permissions

Directory classes	Read	Write	Execute
u = User/Owner	Yes	Yes	Yes
g = Group	Yes	No	Yes

Directory classes	Read	Write	Execute
o = Others	Yes	No	Yes

The default file and directory permissions are specified by the umask variable (refer to 5.3, “umask” on page 94).

Access modes are represented two ways in the AIX operating system, symbolically and numerically:

Symbolic

r = read
w = write
x = execute

Octal

4 = read
2 = write
1 = execute

Permission bits are often written in octal. For example, the following in octal is 754:

```
rwxr-xr--
```

Octal is convenient because the first digit represents the owner’s permissions, the second digit is the group’s permissions, and the third digit is everyone else’s permissions. When using octal notation, sometimes four digits are shown. In this case, the first digit contains the system permissions, which are explained in 5.2.2, “Permissions bits - Advanced” on page 90.

Execute permission

The *execute* permission is not as simple as it might appear.

For binary programs (produced by a compiler, and linked for execution), it functions in the obvious way.

A shell script execution is also limited by read. A shell interpreter must be able to read the commands in order to run them. It is also possible to run an interpreter (shell) with a non-executable file as input if it’s readable. The logic is that the current shell is reading the shell script as a data file; the AIX kernel is not executing the shell script.

The meaning of execute permission for directories is described in 5.2, “File and directory security concepts” on page 86

5.1.7 Changing file permissions

This section discusses the commands related to changing permissions on files mode, owner, and group.

The chmod command

You can modify the read, write, and execute permissions of specified files and modify the search permission codes of specified directories with the `chmod` command.

There are two methods for setting and controlling permission bits: the `chmod` command and the set of access control list (ACL) commands. The ACL commands are primarily for working with extended access list functions (refer to 5.6, “Access control list (ACL)” on page 100). The `chmod` command is the primary tool for changing base permissions bits. They are the read, write, and execute permissions of specified files and modify the search permission codes of specified directories.

The `chmod` operands can be octal (sometimes called *absolute*) or symbolic. Octal notation is common, and this is imbedded in many script programs and shown in most texts. A symbolic operand can do relative changes, such as add (+) and subtract (-) or clear and set (=). An octal operand simply replaces the total permissions value.

For example, to add a type of permission to several files in symbolic format:

```
# chmod g+w chap1 chap2
```

This adds write permission for group members to the files chap1 and chap2.

To use the octal (or absolute) form of the `chmod` command:

```
chmod 644 text
```

This sets read and write permission for the owner, and it sets read-only mode for the group and others. This also removes all extended ACLs that might be associated with the file.

The use of an octal operand will disable the extended ACL parameters (if any) associated with the file. If you use extended ACLs, you must use `chmod` with symbolic operands when working the files containing the extended ACLs. An alternative is to use the ACL editor (discussed in 5.6, “Access control list (ACL)” on page 100). For example, you should use:

```
chmod a+rw myfile
```

Rather than:

```
chmod 644 myfile.
```

This may be an unfamiliar requirement, and it is difficult to remember not to use octal notation. It is almost impossible to enforce the use of only symbolic operands. The `tcbck` command can locate files with disable extended ACLs. See Chapter 6, “Trusted computing base (TCB)” on page 109 for an introduction to the `tcbck` command.

The chown command

You can change the owner of your files with the `chown` command. Optionally, a group can also be changed.

Only the root user can change the owner of a file. You can change the group of a file only if you are a root user or if you own the file. If you own the file but are not a root user, you can change the group only to a group of which you are a member.

When a symbolic link is encountered, and you have not specified the `-h` flag, the `chown` command changes the ownership of the file or directory pointed to by the link and not the ownership of the link itself. If you specify the `-h` flag, the `chown` command has the opposite effect and changes the ownership of the link itself and not that of the file or directory pointed to by the link.

If you specify the `-R` flag, the `chown` command recursively descends the specified directories. If you specify both the `-h` flag and the `-R` flag, the `chown` command descends the specified directories recursively, and when a symbolic link is encountered, the ownership of the link itself is changed and not that of the file or directory pointed to by the link.

Examples:

- To change the owner of the file program.c:

  ```
  # chown joe program.c
  ```

  The user access permissions for program.c now apply to joe. As the owner, joe can use the `chmod` command to permit or deny other users access to program.c.

- To change the owner and group of all files in the directory /tmp/src to owner jane and group build:

  ```
  # chown -R jane:build /tmp/src
  ```

The chgrp command

The `chgrp` command changes the group associated with the specified file or directory to the specified group name or group ID number. When a symbolic link is encountered, and you have not specified the `-h` flag, the `chgrp`

command changes the group ownership of the file or directory pointed to by the link and not the group ownership of the link itself.

Examples:

- To change the group ownership of the file or directory named proposals to staff:

  ```
  chgrp staff proposals
  ```

  The group access permissions for proposals now apply to the staff group.

- To change the group ownership of the directory named proposals, and of all the files and subdirectories under it, to staff:

  ```
  chgrp -R staff proposals
  ```

  The group access permissions for proposals and for all the files and subdirectories under it now apply to the staff group.

5.2 File and directory security concepts

The basic elements of AIX file security are quite simple. The permission bits for the file and the permission bits for the directory containing the file are the key elements. In UNIX, a directory is a type of file. As a file, it has permission bits in its own inode. Permissions to read and write in a directory are independent of permissions to read and write the files named within the directory.

This is a critical concept and is not intuitive to anyone whose background includes DOS, OS/2, or MVS. A user with write permission for a directory can create, rename, or remove a file. Directory execute permission is required to access the directory (when looking for a file, for example).

A file's permission bits are involved when opening, reading, writing, updating, or executing the file. Directory permission is required to find a file before opening it for use.

The owner of a file or directory can always change the permission bits. The owner is usually the user who created the file or directory, but ownership can be transferred (by root) to another user. Three permission bits apply to the owner; the owner can set these to protect themselves against their own mistakes. For example, the owner of a file can set the owner permission bits to "r-x." This will prevent the owner from writing in the file. However, the owner can always change the permission bits for the file (using the `chmod` command) if they really want to write in it.

Directory permissions

Directory permissions are especially important for AIX system files, such as those in /usr. You should not allow users to add files to /usr and its subdirectories (unless there is a good reason for doing so). You control this by not allowing write permission for *others* in any of these directories. You must also be careful not to add users to any groups that have write permission for system directories or files. This is the default condition when you receive AIX. You can list directory permissions with the command:

```
# ls -ld /usr
drwxr-xr-x  33 bin      bin         1024 May 02 11:23 /usr
```

You can also display directory permissions by listing (with the `ls -l` command) the contents of the directory which is one level above the directory whose permissions you want.

A directory cannot be executed, and the “x” permission bit is used to control the ability to search the directory. To `cd` to a directory, or to use it as part of a path name, one must have search permission (x) for the directory. To list the files in a directory, one must have read permission (r) for the directory.

You (an administrator) must care for permission bits in directories because all other security depends on these. Consider the following extreme example:

1. An intruder discovers that the permission bits in the root directory are set to rwxrwxrwx. This means that any non-root user can write into the root directory.
2. The intruder renames /etc to /trash, using the `mv` command. They can do this because they can write in the root directory.
3. The system soon crashes or hangs because all the /etc files have disappeared. The administrator will need to boot from a maintenance CD-ROM or tape and spend time diagnosing and fixing the problem.

While this is not a very useful attack on a system, it illustrates the critical nature of directory permissions.

The administrator must ensure that system directories (at any level in the directory tree) are not writable by non-administrative users. In general, no directory should be writable by the *world (other)*; however, there are exceptions. The /tmp and various lost+found directories are special cases (discussed in 5.8.1, “The /tmp directory” on page 106). As distributed, AIX contains a number of world-writable directories. These can be listed with the command:

```
find / -perm -0007 -type d
```

Some of these world-writable directories use the “sticky bit,” explained in 5.2.2, “Permissions bits - Advanced” on page 90, as an additional control.

Basic directory permission principles

- To use a file (as data or an executable program), the user must have search (that is, execute) permission for all directories in the path. They must also have appropriate permission for the file itself.
- To list a directory (with the `ls` command, for example), a user must have read permission for the directory.
- With write permission for a directory, a user can add new files and subdirectories, move (rename) files, and possibly delete files and subdirectories in the directory (deleting a subdirectory requires permissions to delete all the files in the subdirectory). These actions can be taken regardless of the permissions on the files in the directory. In one way or another, a user can subvert any file, lower-level directory, or files in a lower-level directory if they has write permission to the directory.

As distributed, AIX has all file and directory permissions, owners, and group owners correctly set for secure operation. You should consider the security consequences before changing any of these controls.

5.2.1 The ls command

The `ls` command is probably the single most important command for you as a security administrator. (The `find` command is the second most important.) You must understand the detailed information it displays and learn to use several of its optional flags.

The basic `ls` command displays a list of the files in the current directory and displays nothing else.

For more information, you will normally use one of the following forms:

ls -al
: Displays information about all the files in the current directory, including hidden files (whose names begin with a period).

ls -ld
: Displays information about the current directory itself.

ls -l /some/file/name
: Displays information about a particular file.

ls -ld /some/directory/name
: Displays information about a specific directory.

Interpreting fields reported by the `ls` command is shown in Figure 7 on page 89. The inode number, shown in the figure, is not displayed by the forms of the `ls` command shown here. The `i` flag can be used with it to display inode numbers, but these are not useful in most situations.

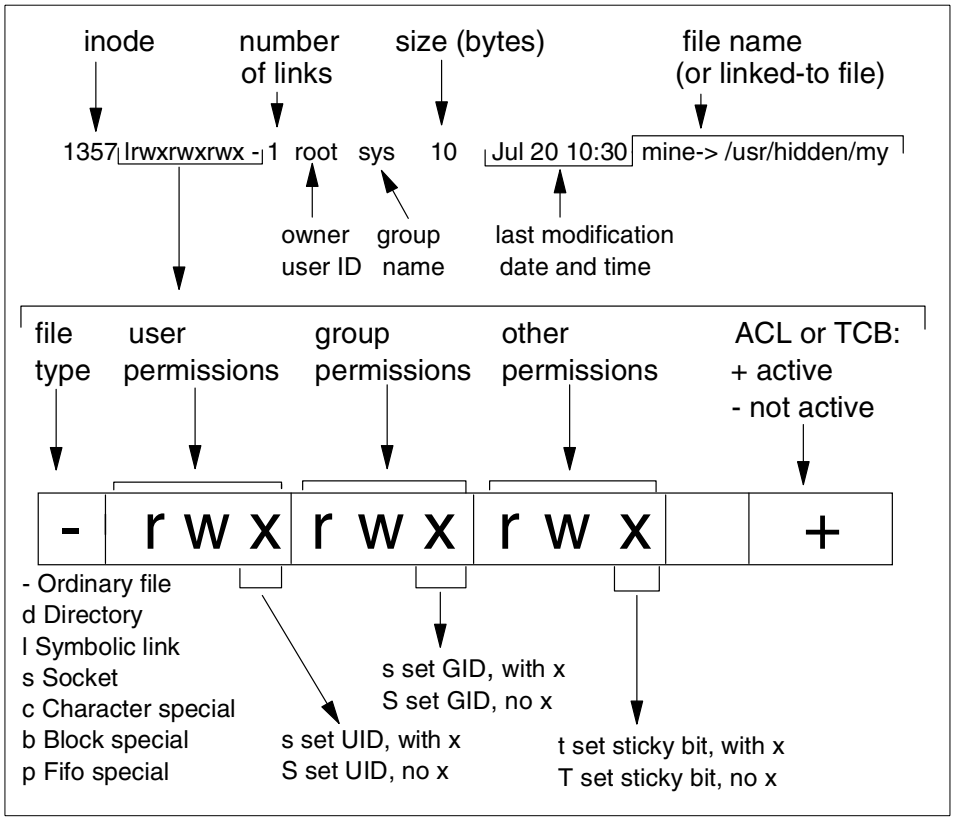

Figure 7. The ls command

The set UID, set GID, and sticky bits are described in detail in 5.2.2, “Permissions bits - Advanced” on page 90.

The owner’s user ID is shown by all the long forms of `ls`. Remember that a file (or directory) owner can change any of the attributes of the file except the owner and group names. Only root can change these. An inode contains the UID and GID of the owner, not the user name or group name. If the UID (or GID) is no longer registered in /etc/passwd (or /etc/group), then the UID or GID is displayed instead of the user name or group name. This is an indication of an unowned file. Refer to 5.1.5, “Unowned files” on page 80.

A great deal of information relevant to security is packed into `ls` output, and you should understand all of it.

5.2.2 Permissions bits - Advanced

UNIX uses 12 permission bits. Of these, nine are the basic r/w/x permissions for user/group/other, and were described in 5.1.6, “File and directory permissions - Basic” on page 81. The three remaining bits are somewhat more complex. They are:

1. The Set-user-ID-on-execution (SUID) bit
2. The Set-group-ID-on-execution (SGID) bit
3. The save-text (or sticky) (SVTX) bit

These bits are set with the `chmod` command, using either symbolic operands or a 4-digit octal operand.

These bits are critical for security controls and are displayed by modifying the normal “rwxrwxrwx” string used to display the basic permission bits. For display purposes, these bits modify the three “x” bits in the normal display.

SUID

SUID on an executable file means that when the file runs, the process runs with an effective UID of the owner of the file. SUID is not supported on shell scripts. SUID has no meaning on a directory.

The SUID bit is displayed by changing the “x” in the user “rwx” to a lowercase “s” if execute is on, or an uppercase “S” if execute is off. For example:

```
# chmod u+s myprog
# ls -l myprog
   -r-sr-xr-x 1 root sys 3254 Jun 1 11:30 myprog
```

This has the SUID bit set. If you (logged into the system as user joe) execute myprog, it will execute with root authority. Since root can bypass almost all security controls, this could be dangerous. The SUID root programs supplied as part of AIX can be trusted to behave correctly. As an administrator, you must be very careful about installing new SUID root programs that anyone can execute.

In this case, myprog might be a copy of the Korn shell (or something similar). By executing myprog (with SUID to root), you effectively become root. You can enter any system command using this shell, and all the commands will run under root’s authority. This situation, a shell with *SUID to root*, is a prime goal of any system intruder.

The SUID bit can be set (using the `chmod` command) only by the owner of a file or by root. It is automatically removed by the `cp` command. There is no direct way for a normal user to create an SUID root file.

The SUID function can be used with owners other than root. It can be used, for example, to ensure that a file is accessed only by a certain program. For example:

```
-rw------- 1 joe eng 5432 Jun 2 13:45 mydata
-r-sr-xr-x 1 joe eng 2345 Jun 1 11:30 myprog
```

This permits anyone to execute myprog. Only user ID joe can access mydata. Since anyone can execute myprog, and since myprog uses SUID to execute as joe, anyone can access mydata only by executing myprog. The assumption is that this program contains whatever security controls it needs to manage proper access to mydata. You should completely understand this example. It incorporates many of the key elements of permission bits and represents a practical way to control the use of shared data.

A typical AIX system has multiple programs that SUID to root. The administrator of a multi-user system should ensure that any additions (new programs that SUID root) are known, trusted programs. AIX provides a utility, tcbck, that can help manage this. It is described in Chapter 6, “Trusted computing base (TCB)” on page 109.

SGID

SGID on an executable file means that when the file runs, the process runs with an effective GID of the group owner of the file.

SGID permission bits are propagated down through the directory structure so that any directory created in a directory with the SGID bit set also inherits that bit.

The SGID bit is displayed by changing the “x” in the group “rwx” to an lowercase “s” if execute is on, or an uppercase “S” if execute is off. For example:

```
chmod g+s <filename>
```

The SGID function works just like the SUID function, using a file’s group identity instead of the owner identity. The SGID bit has a special meaning when used with a directory, where it determines how group ownership for new files is assigned.

For security reasons, AIX ignores the SUID and SGID bits when executing shell scripts. That is, only compiled “object code” programs can SUID to

another UID for execution. Some UNIX systems permit shell scripts to SUID. In principle, this is useful. In practice, it is an endless source of security breaches and has been removed from AIX for this reason.

Permissions (for files or directories) are not cumulative for the owner, group, or other fields. In general, the owner field provides more permissions than the group field, and the group field more permissions than the other field, but this is not required. For example:

```
-r--rw-rwx 1 joe xyz 3210 Jun 3 15:15 mystuff
```

This has rather unusual (but valid) permissions. The owner (joe) cannot write or execute this file. (He can change the permissions, of course, and add more permissions for himself, but, as shown here, he cannot write or execute the file.) Any member of group xyz can read or write the file. Anyone else (other than the owner and any members of the group) can read, write, or execute the file.

Stated a different way, the permissions assigned to the owner, group, or other are also restrictions. The owner, for example, is not considered part of the “other.” This aspect can be used to exclude certain users from accessing a file. This can be done by creating a new group containing all the users to be excluded. The file’s group-owner is then changed to this new group name. The group permissions are set to ‘---’. The result is that no member of the group can access the file, even if the file has full access for the rest of the world.

SVTX
SVTX on a directory means that even if the directory has global write permission (for example /tmp), users cannot delete a file within it unless they are either the owner of the file or directory. SVTX on a file has no meaning in AIX (it was used in earlier versions of UNIX).

The SVTX bit is displayed by changing the “x” in the other “rwx” to an lowercase “t” if execute (x) is on, or an uppercase “T” if execute (x) is off. For example:

```
chmod +t <directoryname>
```

In a normal directory (without the sticky bit), any user with write access can move or remove files in the directory. File permission bits are verified when a file is opened and the commands `mv` and `rm` do not open a file (unless the `mv` is to another file system).

This is a severe exposure with directories, such as /tmp, that are world-writable. When the sticky bit is set, only the owner of a file can delete it,

even if the directory is world-writable. Note that any security information effective in a specific directory is not propagated to lower directories.

This exposure can be prevented by use of the sticky bit. When it is set, only the owner of the directory or the owner of the file in the directory can delete or rename the file, even if “other” has write permission to the directory and file. This is useful for files in /tmp because this directory must have write permission for “other.” Error logs, daily security or accounting reports, and similar files are usually written in /tmp. This use of the sticky bit prevents just anyone from deleting such files. We recommend setting the sticky bit for /tmp (refer 5.8.1, “The /tmp directory” on page 106 for more info on /tmp). However, note that some major software packages may not work (or require special setup) in this case. The sticky bit is set with the `chmod` command.

Directory permission summary

To summarize, permission bits used with directories have the following meanings:

- The SUID bit is ignored in directories.
- The SGID bit is also named the *group inheritance flag* when used with a directory. It controls what group name (actually, what GID) is assigned to new files created in the directory (including new subdirectories). If this flag is set, the GID assigned to the directory itself is used as the GID for any new files created in the directory. If the flag is not set, the GID of a new file is the current group of the user who created the file. (A user can change their current group with the `newgrp` command.)
- The group inheritance function can be set as the default for a file system by defining the grpid parameter in the stanza for the file system in /etc/filesystems or when the file system is mounted.
- The sticky bit means that, even though the directory is writable by the current user, only the owner of a file in the directory can delete a file. (The owner of the directory and root can also delete files.) Note that this flag also prevents non-owners from renaming a file (with the `mv` command). This flag is commonly used for shared directories, such as /tmp, and various spool and mail directories.
- Read permission (in user, group, or other fields) permits a user to read the directory (but not the files within the directory). The `ls` command, among many others, reads directories. It will not list a directory unless the current user has read access to the directory. Read permission in a directory is required in order to use wild cards when referencing the directory.
- Write permission (in user, group, or other fields) permits a user to add, delete, or change entries in the directory. A file can be added, deleted, or

renamed. An existing file cannot be read or written (unless the user has appropriate permission to the file), but it can be deleted or renamed since these actions take place in the directory, not the file itself. This is why directory write permissions are so important for security administration.

- Execute permission is called search permission when applied to a directory. It permits the directory to be used as part of an explicit path name. To access the /u/mydir/file3 file, the caller must have search access to the root directory, the u directory, and the mydir directory. Search permission does not permit listing or reading the whole directory; it permits use of a single entry in the directory. The user must, by some external means, know the name of the entry (that is, the path name) of the file they want.

Table 8 summarizes file and directory permissions.

Table 8. Permissions summary

Permission bit	File	Directory
r	User can read the contents of file.	User can list the contents of the directory.
w	User can change the contents of file.	User can create and remove files within directory.
x	User can use the file name as a command.	User can cd to a directory and can use it in path.
SUID	Program runs with effective UID of owner.	Ignored.
SGID	Program runs with effective GID of owner.	Files created in directory inherit the same group as the directory.
SVTX	Ignored.	To delete a file in directory, the user must own the file or directory.

5.3 umask

Every file (and directory) has permission bits. The owner can change them with the `chmod` command. The initial, default permissions set when a file is created are controlled by a parameter named *umask*. For reasons going back to the early days of UNIX, the umask value is used in an odd way. Default permissions are established by assuming permissions (“rwxrwxrwx” or octal 777 for directories, and “rw-rw-rw-” or octal 666 for normal files) and removing

the permission bits specified in the umask (which is always expressed in octal).

The default umask is 022 (octal). Therefore, default permissions are:

- 777 removing 022 = 755 = rwxr-xr-x (for a directory)
- 666 removing 022 = 644 = rw-r--r-- (for a file)

The umask parameter can be changed by the user with the `umask` command (which is a shell command). There is no way to enforce a standard value for users. The default values are suitable for most uses. A different default can be set by placing the `umask` command in a user's $HOME/.profile file, for example. However, the user can change the value at any time. A user's initial umask value can be set through SMIT.

Type `umask` to see the current setting.

To change the umask setting, type `umask <number>`, as in:

```
umask 022
```

5.4 User and system limits on stand-alone systems

Standard AIX has facilities to limit the extent to which a system administrator can deny service to users. User limits allows the root user to implement appropriate physical security restrictions to prevent unauthorized users from disabling the system.

The following sections can help prevent certain types of “denial of service attacks”.

5.4.1 ulimit

A system can be paralyzed in a variety of ways, for example, a self spawning process can consume large amounts of CPU time, which could cause high paging activity, thus slowing the system down. This can be prevented by setting limits against a user's system usage.

There are two types of user limits: hard limits and soft limits. The limits for all users on the system including the default are held in /etc/security/limits.

The *hard limits* are generally set by the system administrator as a maximum limitation on a user's processes.

To view the current hard limits:

```
ulimit -Ha
```

The *soft limits* are the values directly used by the kernel to limit a processes system resources.

To view the current soft limits:

```
ulimit -a
```

These values are used as default settings when a new user is added to the system. The values are set with the `mkuser` command when the user is added to the system, or changed with the `chuser` command. The following is an example of part of the `smit mkuser` fast path:

```
Soft FILE size                          [2097151]               #
Soft CPU time                           [-1]
Soft DATA segment                       [262144]                #
Soft STACK size                         [65536]                 #
Soft CORE file size                     [2097151]               #
Hard FILE size                          []                      #
Hard CPU time                           []
Hard DATA segment                       []                      #
Hard STACK size                         []                      #
Hard CORE file size                     []                      #
```

Once a hard limit has been decreased by a process, it cannot be increased without root privilege, even to revert to the original limit. A user can decrease their limit value at anytime. Any changes made by a user are temporary and will be lost upon logout. Any changes made by *root* or a member of the *security* group are permanent. This can be set by the `ulimit` command or via SMIT.

The `ulimit` command sets or reports user process resource limits as defined in the /etc/security/limits file. This file contains defaults limits as shown in Table 9.

Table 9. System defaults

Type	Soft Limit	Hard Limit	Description
File Size (blocks)	2097151	-1	A user cannot write to a file larger than the limit.
Core File Size (blocks)	2097151	-1	Will not allow a user to create a core file larger than the set SOFT limit.
CPU (units in seconds)	3600	-1	Not enforced by the kernel.
Data Segment (blocks)	262144	-1	Data is the largest data segment allowed.

Type	Soft Limit	Hard Limit	Description
Stack Segment (blocks)	65536	-1	Maximum size of the stack segment for a process.
Resident Set (blocks)	65536	-1	Not enforced by the kernel.
nofiles	2000	32767	Number of open files.
A negative value denotes “unlimited”. One block = 512 bytes			

File size 2097151 is approximately 1GB of disk space. We recommend a smaller value in a multi-user environment. For example, 16,384 (which allows file creation up to 8 MB) might be a reasonable value. The smallest number that can be set through SMIT appears to be 8192. The largest number the limit can be set to is 4194303 512-byte blocks, or 2 GB. The term *unlimited*, a value of “-1”, should be used in cases where you want to create files larger than 2 GIG in size.

Setting the default limits in the /etc/security/limits file sets system-wide limits, not just limits taken on by a user when that user is created.

For more information about user and system resource limits, refer to the getrlimit, setrlimit, or vlimit subroutines in *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 1*, SC23-4159.

5.4.2 Disk quotas

A *disk quota system* provides a method for controlling disk space usage for AIX JFS only. Denying a user disk space can prevent problems occurring, such as a file system becoming full on disk.

Disk quotas should be used only for file systems that contain users’ home directories and files. It is recommended that disk quotas never be assigned to the /tmp directory.

A system administrator would consider implementing the disk quota system under the following conditions:

- The system has limited disk space.
- More file system security is desired.
- Disk usage levels by users are large.

To set up the disk quota system, you must have root authority. There are three steps to setting up disk quotas:

1. Enable the file system to be protected for quotas.

   This can be done with the `chfs` command or editing /etc/filesystems:

```
# chfs -a quota=userquota /home
# vi /etc/filesystems
/home:
             dev       = /dev/hd1
             vol       = "/home"
             mount     = true
             check     = true
             free      = false
             vfs       = jfs
             log       = /dev/hd8
             quota     = userquota,groupquota
             options   = rw
#
```

   Mount the file systems if not already mounted.

2. Set up the quota limits for a user by using the `edquota` command:

```
# edquota -u joe
/home blocks in use 30 limits (soft = 100 hard = 150)
inodes in use 73 limits (soft = 200 hard = 250)
```

   The `-u` flag uses the default editor as set by the environmental variable EDITOR.

3. Turn on quota monitoring of the system.
   - To enable the quota subsystem: `quotaon /home`
   - To check the consistency of the quota files against the disk usage: `quotacheck`
   - To check on the usage of the allocated quota: `repquota`

We recommend that you do `quotacheck` each time you first enable quotas on a file system and after you reboot the system.

To enable this check and to turn on quotas during system startup, add the following lines at the end of the /etc/rc file:

```
echo " Enabling filesystem quotas "
/usr/sbin/quotacheck -a
/usr/sbin/quotaon -a
```

There are three stages to quota limits:

Soft limits

Determines the point to which the user can keep adding data to the file system.

Grace period

When the soft limit is passed, it enters the *grace period*. The default for the grace period is seven days. The user can keep adding data to the file system until they reach the hard limit, then no more data can be stored.

Hard limit

If the user is still above the soft limit when the grace period expires, the hard limit drops to the soft limit and no more data can be stored.

5.5 File timestamps

UNIX systems, including AIX, maintain three timestamps for files (including directories). These can be important for resolving security questions. The time stamps are:

atime This is the time the file was last accessed. In effect, this is the last time the file was opened. For example, a user could vi the file but not make any changes, yet the timestamp for the file would be updated.

ctime This is the last time the inode for the file was changed. (It is not the creation time, unless file creation was the last event for the inode.) The inode is changed when permissions are changed, the owner is changed, the file size (number of clusters) is changed, and so forth.

mtime This is the last time the contents of the file were changed. This generally means the file was opened for output. This time can easily be manipulated by the file owner or root with the `touch` command. For example, the command `touch 0101000095 afile` will set the mtime to Jan 1, 1995. Only write authority to the inode is needed to manipulate the times.

The long forms of the `ls` command will list the file timestamps. For example:

ls -ul This lists the atime.

ls -cl This lists the ctime.

ls -l This lists the mtime.

The `find` command can be used to reference all three time stamps.

5.6 Access control list (ACL)

AIX has an additional security function for files. This is the access control list (ACL) facility.

AIX ACLs can provide much finer-grained access control than can be obtained with permission bits. As a general case, explicit ACL control is not normally used with workstations. It may be used within specific applications on servers. That is, normal AIX usage typically does not involve individual users randomly assigning ACLs as the mood strikes them. (Although it is possible and there are no controls to prevent it.) An example of a typical usage would be a planned set of ACLs for the payroll department's files.

The major task in administering access control is to define the group memberships of users, because these memberships determine the users' access rights to the files they do not own.

The ACL for a file cannot exceed one memory page (approximately 4096 bytes) in size.

ACL are maintained by the `aclget`, `acledit`, and the `aclput` commands.

Note

Although the `chmod` command in numeric mode (with octal notations) can set base permissions and attributes, the `chmod` subroutine, which the command calls, disables extended permissions. If you use the numeric mode of the `chmod` command on a file that has an ACL, extended permissions are disabled. The symbolic mode of the `chmod` command does not disable extended permissions. For information on numeric and symbolic mode, refer to 5.1.6, "File and directory permissions - Basic" on page 81.

Base permissions

Base permissions are the traditional file access modes assigned to the file owner, file group, and other users. The access modes are read (r), write (w), and execute/search (x).

In an ACL, base permissions are in the following format with the *mode* parameter expressed as rwx (with a dash replacing each unspecified permission):

```
attributes: SUID, or SGID or SVTX in any combination
base permissions
    owner(<user>):  rwx
```

```
    group(<group>):  r-x
    others:  r-x
extended permissions
    disabled
```

Extended permissions

Every file and directory has a “base ACL” because the standard permission bits are also the base ACL. The extended ACL functions are usually simply called the ACL functions. Extended permissions allow the owner to define access to a file more precisely. Extended permissions extend the base file permissions (user, group, others) by permitting, denying, or specifying access modes for specific individuals, groups, or user and group combinations. Any user can create an extended ACL for a file they own.

The permit, deny, and specify keywords are defined as follows:

permit Grants the user or group the specified access to the file.

deny Restricts the user or group from using the specified access to the file.

specify Precisely defines the file access for the user or group.

If a user is denied a particular access by either a deny or a specify keyword, no base permission or general extended permission can override that denial.

When both a user and group are defined in an extended permission, only the specific user and group combination receives the access. There is an “and” relation between the elements in a list.

The enabled keyword must be included in the access control information for the extended permissions to take effect. The default value is the disabled keyword. Using `chmod` with an octal operand is one way to set the disabled state.

In an ACL, extended permissions are shown as follows:

```
extended permissions:
enabled | disabled
permit Mode UserInfo....:
deny   Mode UserInfo....:
specify Mode UserInfo....:
```

Use a separate line for each `permit`, `deny`, or `specify` entry. The Mode parameter is expressed as rwx (with a dash replacing each unspecified permission). The `UserInfo` parameter is expressed as u:UserName, or

g:GroupName, or a comma-separated combination of u:UserName and g:GroupName.

Note

If more than one user name is specified in an entry, that entry cannot be used in an access control decision, because a process has only one user ID.

An example of an ACL is as follows:

```
attributes: SUID
base permissions:
owner(frank): rw-
group(system): r-x
others: ---
extended permissions:
enabled
permit rw- u:jane
deny r-- u:katie, g:system
specify r-- u:john, g:gateway, g:mail
permit rw- g:account, g:finance
```

The parts of the ACL and their meanings are as followings:

- The first line indicates that the setuid bit is turned on.
- The next line, which introduces the base permissions, is optional.
- The next three lines specify the base permissions. The owner and group names in parentheses are for information only. Changing these names does not alter the file owner or file group. Only the `chown` command and the `chgrp` command can change these file attributes.
- The next line, which introduces the extended permissions, is optional.
- The next line specifies that the extended permissions that follow are enabled.
- The last four lines are the extended entries. The first extended entry grants user jane read (r) and write (w) permission on the file.
- The second extended entry denies read (r) access to user katie when she is a member of the system group.
- The third extended entry specifies that as long as user john is a member of both the gateway group and the mail group, he can have read (r) access. If user john is not a member of either one of these groups, this extended permission does not apply.

- The last extended entry grants any user in both the account group and the finance group read (r) and write (w) permission.

Note

More than one extended entry can be applied to a process, with restrictive modes taking precedence over permissive modes.

The meaning of an ACL can become complex for a user who is a member of multiple groups. An ACL might include entries for several of the user's groups, and these may conflict. For example, a user may belong to GROUP1 and GROUP2. A given ACL may provide read access for GROUP1 and execute access for GROUP2. These conflicts are resolved in this order:

1. If a specify operand exists for any of a user's groups (or for their own user ID), the specify will set a maximum access level. If multiple specifies exist (for different groups and/or the user ID), the least-common denominator of all the specifies is used. Access rights will never be higher than this, and may be less, due to deny permissions.
2. All (positive) access permissions (for the user and all of the user's groups) are added together.
3. All deny (negative) access permissions (for the user and any of the user's groups) are then subtracted. The result is further limited by specify restrictions (if any).

A deny function, in a sense, is more powerful than permit functions because a single deny can override any number of permits. This result may surprise users and administrators, but it is a logical result of ACLs, denies, and many-group operation. If a user is unable to access a file, and you cannot understand why, you should check the ACL for any deny associated with group IDs. The user may be a member of the denied group. The same effect can be caused by group-level specifies.

The ACL commands referenced here are primarily for extended ACL functions, but they can also be used instead of `chmod` to control base permission bits. The commands are:

aclget Gets the ACL for a file.

aclput Sets the ACL for a file.

acledit Combines aclget and aclput.

The aclget command

You can display the access control information of a file with the `aclget`

command. The information that you view includes attributes, base permissions, and extended permissions.

For example, to display the access control information for the file *status* enter:

```
aclget status
```

The aclput command

You can set the access control information of a file with the `aclput` command.

The `-i` flag can be used to obtain input from a file rather than from system input. If the access control information in the file specified by the InFile parameter is not correct when you try to apply it to a file, an error message preceded by an asterisk is added to the input file. For example:

- To set the access control information for the status file with information stored in the *acldefs* file, enter:

  ```
  aclput -i acldefs status
  ```

Another way of doing this is by using the pipe command. For example:

- To set the access control information for the status file with the same information used for the plans file, enter:

  ```
  aclget plans | aclput status
  ```

The acledit command

You can change the access control information of a file with the `acledit` command. The command displays the current access control information and lets the file owner change it. Before making any changes permanent, the command asks if you want to proceed.

Note

The EDITOR environment variable must be specified with a complete path name; otherwise, the `acledit` command will fail. This is partly to avoid security exposures due to improper PATH parameters. For example: EDITOR=/usr/bin/vi.

If the `acledit` command is operating in a trusted path, the editor must have the trusted process attribute set.

For example, to edit the access control information for the plans file, enter:

```
acledit plans
```

Before making any changes permanent, the command asks if you want to proceed. If you have entered incorrect information, for example, an invalid group, an error will appear under the incorrect entry line.

Do not depend on extended ACLs in heterogeneous networks, since non-AIX systems will ignore them. Only AIX systems will observe extended ACLs over a network. For reasons of compatibility, we recommend that you discourage use of the ACL functions except for pre-planned uses with major applications.

5.7 Device files

The system administrator should review permissions on device files. Insecure permissions on device files allows direct access to hardware devices and their kernel data structures. As distributed by AIX, the default permissions for device files are listed in Table 10.

Table 10. Default device file permissions

Permissions - Octal	Permissions - Symbolic	Device
666	rw-rw-rw	Any TTY device not in use
664	rw-rw-r--	Any TTY device in use
660	rw-rw----	hdisk# devices
600	rw-------	Console
440	r--r-----	Mem, kmem, and pmem
666	rw-rw-rw	Keyboard
666	rw-rw-rw	Tape drive
666	rw-rw-rw	Diskette drive
444	r--r--r--	CD-ROM drive

5.8 AIX error logging

Security exposures sometimes happen because of errors. AIX has a good error logging and reporting facility that reports hardware and software problems. You should list the error log regularly.

The `errpt` command is used directly, or with SMIT.

Examples of `errpt` commands are as follows:

- To display a complete summary report, enter:

```
errpt | pg
```

- To display a complete detailed report, enter:

```
errpt -a | pg
```

- To display a detailed report of all errors logged for the error identifier E19E094F, enter:

```
errpt -a -j E19E094F
```

- To display a detailed report of all errors logged in the past 24 hours, enter:

```
errpt -a -s mmddhhmmyy
```

Examples of clearing the error log commands are as follows:

- To delete all entries from the error log, enter:

```
errclear 0
```

- To delete all entries in the error log classified as software errors, enter:

```
errclear -d S 0
```

Only the root user can run this command. The `errclear` command clears the specified entries, but does not decrease the error log file size.

We recommend that error logging should always be active. It is active as long as errdemon is running, and this is started automatically when the system is booted.

5.8.1 The /tmp directory

Many programs create and use work files in the /tmp directory. This is not very secure since this directory is not protected, and anyone can read these files. The line TMP=$HOME/tmp in a user's $HOME/.profile may help. The user should create the subdirectory *tmp* in their home directory. Some packages and commands use the TMP environmental variable to place temporary files.

Many applications place files in the /tmp directory. Some applications fail to delete these files when they end. Part of the system administrator's routine should be to check /tmp regularly and delete those files that have not been accessed within the past few days. Fortunately, `skulker` performs this task on a daily basis and (unless it is disabled) the care of the /tmp directory can be largely left to it.

The normal mode for /tmp is that all users can read all files in the directory and all users can create and write new files, but only the owner of a file can delete it. This is done by setting other (or world) access to read and write and

turning on the sticky bit (SVTX). However, some applications may require that the sticky bit for the directory be removed, thus allowing complete read/write/delete access to the directory for all users. This should be avoided if possible, but if it cannot, then all users should be made aware of the potential for damage if files are stored in the /tmp directory.

Confidential, sensitive, or essential data should never be kept in the /tmp directory since this directory is readable by all users. This is not easy to manage since many program packages automatically place work files in /tmp.

If space is required in /tmp, then (provided your applications do not have essential files there, which they should not) it is safe to delete any files in the directory to create space. That is, it is safe from an operating system point of view; you may have unhappy users, but they should not leave useful data in /tmp.

5.8.2 The virscan command

The `virscan` command is designed to detect many common computer viruses. It scans executable files, looking for signatures of viruses known when this version of the program was made available.

The main use of the `virscan` command is for AIX systems that are involved with PC systems since it was originally developed to detect the presence of known computer viruses in PC systems. It was adapted for use in an AIX environment but, at this time, contains no known AIX virus signatures. The virus signature file contains only known PC-based virus signatures since it is there to search for PC-based viruses on AIX systems, not for AIX-specific viruses.

The `virscan` command reports of any detected virus but does not attempt to remove viral infections from a system.

The `virscan` command cannot find virus signatures in files that are compressed or encrypted. This includes files that have been compressed by archiving programs. To scan such files, unpack them first and then scan their constituent files.

On systems with damaged directory trees, the `virscan` command terminates with an error. Though not caused by the `virscan` command, this condition will prevent the `virscan` command from scanning the disk correctly. Run the `fsck` command to diagnose any error conditions.

Examples:

- To scan all files in the /usr file system, enter:

  ```
  virscan -a /usr
  ```

- To scan all files in the /usr file system and put the names of any infected files into the file positive.vir in the current directory, enter:

  ```
  virscan -a -p /usr
  ```

- To scan the files listed in the files.dat file for viral signatures, enter:

  ```
  virscan -lfiles.dat
  ```

- To scan the /usr file system using the signatures in both the mysig.dat and virsig.lst files, enter:

  ```
  virscan -s/usr/lib/security/scan/virsig.lst -smysig.dat /usr
  ```

- To scan an entire system, enter:

  ```
  virscan /
  ```

In order for the `virscan` command to run, you will need to have the bos.compat.cmds fileset installed.

Chapter 6. Trusted computing base (TCB)

TCB is a good tool to detect penetrations and configuration changes. TCB stores information about files, which can later be used to verify that the files have not been modified.

6.1 Defaults

TCB is not installed by default. You have the option to install TCB during the initial installation. It cannot be added without reinstalling AIX.

You can do a “Preservation Install” and include TCB. However, if you have done any customizing in rootvg, this may remove your changes. Always do a backup of your system before you try this. We cannot guarantee that Preservation Install will keep all your changes, since Preservation Install does not preserve everything. Try it out on a test system if you can. Also refer to Chapter 2, “Preservation Install” in *AIX Version 4.3 Installation Guide*, SC23-4112.

TCB monitors over 600 files, plus the devices (/dev), by default. It stores these files in an ASCII file, /etc/security/sysck.cfg. Make a backup of this file to a floppy disk and write protect it immediately.

6.2 Customizing TCB

The `tcbck` command itself is fairly straight forward. It has three basic operations: add, remove, or check a file in the TCB. For any given file, TCB has the option of monitoring owner, group, permissions including access control list (ACL), hard links, soft links, size, target, and a source file. Some of these options can be calculated from the file when adding it to TCB. For these options, you do not have to specify a value. Here is an explanation of the options:

acl This is the ACL for the file. If this does not match the actual file acl, the `tcbck` command reports/sets the ACL to this value. *Calculated*.

checksum This is the checksum of the file. The value is the output of the `sum -r` command, including spaces. *Calculated*.

class This is the name of a group of files. This attribute allows several files with the same class name to be checked by specifying a single argument to the `tcbck` command. More

then one class can be specified, with each class being separated by a comma.

group This is the GID or name of the file's group. If this does not match the file group, the `tcbck` command reports/sets the GID of the file to this value. *Calculated*.

links These are the hard links to this file. This value must be a comma separated list of absolute path files ("path[,path...]").

mode This is a comma-separated list of values. The allowed values are SUID, SGID, SVTX, and TCB. The file permissions must be the last value specified either as an octal (for example, 755) or as a 9-character string (for example, rwxr-xr-x). If this does not match the actual file mode, the `tcbck` command reports/sets the mode to this value ("SUID,SGID,SVTX,TCB,555"). *Calculated.*

owner This is the UID or name of the file owner. If this does not match the file owner, the `tcbck` command reports/sets the owner ID of the file to this value. *Calculated*.

program This is the associated checking program for the file. The absolute path must be specified. If flags are required, the value should be "path,flag".

size This is the size of the file in bytes. The value is a decimal number. *Calculated*.

source This is the source for the file. If no value is specified, an empty file of the appropriate type is created. The value must be an absolute path name. With this option, TCB attempts to copy the source file over the original before it compares the other values (see Table 12 on page 112). Some security conscious people can put their source files on a CD-ROM, a read-only file system, a write-protected tape or floppy, or a file system that is only mounted when `tcbck` is running. Use caution here, because you may update a program and lose all your changes if you do not update the source file.

symlinks This is the symbolic link to this file. This value must be a comma-separated list of absolute path files ("path[,path...]").

target This is used only for files that are symbolic links. It tells TCB where the file should be linked from.

type This is the type of file. If no value is specified, the command computes a value, which can be one of the following: FILE,

DIRECTORY, FIFO, BLK_DEV, CHAR_DEV, or MPX_DEV.
Calculated.

See also *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126.

6.2.1 TCB add

If you have a custom script or program that you want to make sure no one changes without your knowledge, you can add it to TCB:

```
tcbck -a <full path>/<file> owner group checksum mode acl size type class=custom
```

This command adds (-a) the specified file (</full path>/<file>) to the TCB configuration file /etc/security/sysck.cfg. TCB calculates its owner, group, checksum, mode, ACL, size, and type.

For example, you want to have a copy of `ksh` (not a link) called `rksh` (restricted shell). You want this to be a copy so you can do regression testing when you patch your ksh. You also have an application that expects `rksh` to be in the directory /usr/local.

```
# cp -p /usr/bin/ksh /usr/bin/rksh
# tcbck -a /usr/bin/rksh acl checksum class=usershell,custom \
group mode=TCB,555 owner size source=/usr/bin/ksh \
symlinks=/usr/local/bin/rksh type
```

Note

TCB cannot calculate the values for a file that does not exist, so `tcbck -a` will fail if a file does not exist unless you specify what you want the value to be.

This could also have been done like this:

```
tcbck -a /usr/bin/rksh group=bin mode=TCB,555 owner=bin \
size=240326 source=/usr/bin/ksh type=FILE checksum="$(cat \
/usr/bin/ksh | sum -r)" symlinks=/usr/local/bin/rksh type=FILE \
class=usershell,custom acl="attributes:\nbase permissions\n \
owner(bin):  r-x\ngroup(bin):  r-x\nothers: r-x\n \
extended permissions\ndisabled"
```

Although this command does not require you to copy `ksh` to `rksh`, it is considerably more complex. We recommend creating your file, setting it up like you want it in production, then adding it to TCB. Saving your commands to a file in /etc/security makes updating the TCB entry later much easier.

To prevent TCB check from having problems recognizing the symbolic link, the link also needs to be added to TCB, because tcbck -[ntyp] tree thinks that the symbolic link /usr/local/bin/rksh should not exist and it wants it deleted. tcbck -[ntyp] /usr/bin/rksh thinks that the symbolic link is missing and tries and create it. To prevent this contradiction, you must add /usr/local/bin/rksh to TCB:

```
tcbck -a /usr/local/bin/rksh target=/usr/bin/rksh type=SYMLINK
```

Also, you will note that we specified the mode for the file. This is so tcbck -[ntyp] tree will identify unauthorized links. We will talk more of that in the next section. If the file exists, you only needed to specify “type” not “type=SYMLINK”, since this can be calculated.

Now `rksh` has been added to your TCB configuration. We will use this to run our TCB check.

6.2.2 TCB check

TCB is installed, and you even added your custom programs to it. This, however, will not do you any good unless you check it periodically. See 6.3.1, “How often to check for intruders” on page 118 for more details.

TCB can check an individual file (file name), a class of files (class name), all the TCB files (ALL), or all the files on the system (tree). Checking all the files on the system is the most thorough, and the most time consuming.

The check is divided into four options as shown in Table 11.

Table 11. tcbck checking options

Option	Report	Take Action
-n	yes	no
-t	yes	prompt
-y	yes	yes
-p	no	yes

To see how TCB wants to fix discrepancies, see Table 12.

Table 12. Fixing TCB discrepancies

Discrepancy	TCB actions
checksum	Disables the file by clearing its ACL.

Discrepancy	TCB actions
links	Creates missing hard links. If mode includes TCB, deletes non-authorized links to the files.
program	Invokes the full-path program (which must exist). A message is printed if an error occurs, but no additional action is taken.
size	Disables the file by clearing its ACL.
source	Deletes the file and copies the source file to the file specified. Your i-node will change, so your hard links will no longer be links, but if you are monitoring them, they will be reported and fixed.
syminks	Creates any missing symbolic links. If mode includes TCB, deletes non-authorized symbolic links to the file. TCB bit must be part of the mode for this to work.
target	Deletes the unauthorized link and creates the correct link.
type	Disables the file by clearing its ACL *and* stops any further checks.

Note

Automatically fixing things in TCB is dangerous. We recommend that you do not automatically fix everything (with ALL or tree). Instead, use the interactive flag if you must use tree (to find links, for example), for example:

```
tcbck -t tree
```

After adding files to TCB, you now want to check them:

```
# tcbck -t /usr/bin/rksh
3001-067 Create the file /usr/bin/rksh from the source location? (yes, no) yes
3001-087 The symbolic link from the file /usr/bin/rksh
         to /usr/local/bin/rksh does not exist.
3001-066 Create the link for /usr/local/bin/rksh? (yes, no) yes
```

It looks like you forgot to create /usr/local/bin/rksh, but that's okay, TCB just created the link, and you went ahead and copied the source over too, just to make sure you had the right file.

Also, if the file does not have the TCB bit set in the mode, it will only report extra links without giving you the name and not do anything about them. With the TCB bit set, TCB will give you the names.

Without the TCB bit set:

```
# tcbck -n ALL
3001-041 The file /usr/local/bin/junk has too many links.
3001-043 Use the tree option to find extra links.
```

```
# tcbck -n tree
3001-041 The file /usr/bin/rksh has too many links.
#
```

With the TCB bit set:

```
# tcbck -n ALL
3001-041 The file /usr/bin/rksh has too many links.
3001-043 Use the tree option to find extra links.
# tcbck -n tree
3001-041 The file /usr/bin/rksh has too many links.
3001-032 The link from the file /usr/bin/rksh.test1
         to /usr/bin/rksh should not exist.
3001-089 The symbolic link from the file /usr/local/bin/rksh.test2
         to /usr/bin/rksh should not exist.
```

Unfortunately, you must specify `tree` if you want TCB to find and fix links. When checking links, if automation is on, it will delete them. If you specify `-t` (prompt me), it will ask you if you want to delete them. If you say no, it will ask if you want to add the link to TCB:

```
# tcbck -t tree
3001-041 The file /usr/bin/rksh has too many links.
3001-032 The link from the file /usr/bin/rksh.test1
         to /usr/bin/rksh should not exist.
3001-069 Remove the file /usr/bin/rksh.test1? (yes, no) no
3001-095 Add the new link for /usr/bin/rksh.test1? (yes, no) yes
3001-089 The symbolic link from the file /usr/local/bin/rksh.test2
         to /usr/bin/rksh should not exist.
3001-069 Remove the file /usr/local/bin/rksh.test2? (yes, no) yes
```

Unfortunately, TCB does not have an automated way of removing authorization links from a TCB file. Therefore, if you want to remove authorization from a link for a TCB file, you need to update TCB.

6.2.3 TCB update

You added a link to /usr/bin/rksh in TCB, but now you do not want it anymore:

```
# tcbck -a /usr/bin/rksh link=""
# tcbck -t tree
3001-041 The file /usr/bin/rksh has too many links.
3001-032 The link from the file /usr/bin/rksh.test1
         to /usr/bin/rksh should not exist.
3001-069 Remove the file /usr/bin/rksh.test1? (yes, no) yes
```

If you saved your add commands in a file, you can simply modify them. We recommend keeping a file that looks like this with an entry for each file you added to TCB:

```
#!/bin/ksh
# We accept not responsibility for this program, and offer no
warranties.
# (lawyers....)

# Generated: acl, checksum, group, mode, owner, size, type.
# Specified: class, links, program, source, symlinks

##
# tcbck -a /usr/bin/rksh acl checksum group mode owner size type \
# class=custome,shells source=/usr/bin/ksh
#tcbck -a /usr/bin/rksh acl checksum class=usershell,custom \
#group mode=TCB,555 owner size source=/usr/bin/ksh \
#symlinks=/usr/local/bin/rksh type links=""
```

This allows you to keep a record of the changes you have made to TCB and will help you do updates later. After changing the links for rksh, all that needs to be done is un-comment the lines from the script, run it, then re-comment the lines. This will force TCB to update the stanza with the new size, acl, checksum, and so on.

The `installp` command automatically updates the TCB when you install PTFs. However, E-Fixes, naturally, do not update TCB. So if you apply an E-Fix to your system, you will need to manually update TCB.

Because paranoia is paramount in security, we recommend running a `tcbck -n tree` before and after installing anything (especially PTFs). This verifies the integrity of your system (or the hacker was extremely slick and updated TCB in the hackers installation program). Do not forget to update your floppy copy after you do an update.

Complete the following system update steps:

1. Check your system:
   - Run `tcbck -n tree`.
   - Investigate the output, and fix everything.
2. Remove the write protection on your TCB backup media (/dev/fd0 in this example).
3. Backup the TCB configuration file in one of the following ways:
   - `tar cf /dev/fd0 /etc/security/sysck.cfg`

- doswrite /etc/security/sysck.cfg SYSCK.CFG
- Other ways to do this (backup, cpio, and so forth)

4. Write protect your TCB backup media to prevent someone from destroying or manipulating it.
5. Install your software updates:
   - smit update_all
   - smit install_latest
   - smit installp
6. Check your system:
   - Run tcbck -n tree
   - Investigate the output, and fix everything.
7. Compare TCB with old TCB:
   - Backup the new TCB file:
     - cp /etc/security/sysck.cfg /etc/security/sysck.cfg.NEW
   - Restore the old TCB file:
     - tar -xf /dev/fd0
     - dosread SYSCK.CFG /etc/security/sysck.cfg
     - Whatever method you used to back up your data, use to restore it.
   - Compare the changes on your system: tcbck -n tree (do not update unless you suspect foul play).
8. Put back the new TCB configuration file:
   - mv /etc/security/sysck.cfg.NEW /etc/security/sysck.cfg
9. Backup your new TCB configuration file as you did in steps 2 to 4.

6.2.4 TCB delete

You know your users are never satisfied with the system and its applications, so AIX allows you to clean up your TCB. For example, your users want a menu program instead of rksh. After it is installed, you can add it to TCB and remove rksh. If you have been saving your commands, be sure to remove rksh from your file, or note that you are not using rksh any more:

```
tcbck -d /usr/bin/rksh /usr/local/bin/rksh
rm /usr/bin/rksh /usr/local/bin/rksh
```

6.2.5 The tsh shell

The tsh shell is a good security tool. It only allows you to run programs that are in the TCB and have the TCB mode set.

tsh only checks for the TCB mode, not any of the other values in TCB. So, if a hacker modified /usr/bin/su, it will still run. But a $HOME/su (created by a hacker), which is not in TCB, will not run. tsh depends on tcbck to catch and fix modified programs.

You would have to detect that it changed (by running `tcbck`) and update its ACL (changing its permissions to 000 or ---------).

The user may see you run `tsh`, and create this Trojan tsh, so you have to specify the full path (`/bin/tsh`). You need to run `tcbck` regularly. When you run it, you need to make sure you are running the real `tcbck`. Specify its full path.

If someone penetrates your system, and compromises root, the game is nearly over. If they figure out that you are running `tcbck`, they can modify your TCB to include their exploits. If however, you keep your TCB on a write protected floppy disk, and you restore that every time you run a TCB check, you have a chance. The perpetrator will need to have physical access to your machine to update that disk. If someone has physical access to your machine, then the game is over. It is very important then to keep your machine physically secure, and to have your TCB backed up on write protected media.

6.2.6 Secure Attention Key (SAK)

SAK is a another tool to help protect you from Trojan Horse login programs. However, SAK only works on locally-attached devices. It does not work on network sessions. On a network machine (for example, a PC doing ssh or telnet), you can simply exit the software and restart it. On network connections, you cannot know if users are doing some sort of logging with their terminal emulators, or on their network adapters. Therefore, *never* use a workstation that you do not completely trust to connect remotely to your machines. That trust must be for the hardware, software, and the network segments between you and the server.

SAK does not use enough resources to hurt anything, so we recommend using it. Add a line to /etc/security/login.cfg to the “default” stanza:

```
sak_enabled=true
```

6.3 Planning your system

Here are some questions you should answer when planning your system.

6.3.1 How often to check for intruders

There are different things to consider to determine how often to check your system for modified files. How busy is the system? How many administrators are there? How many files will be monitored? What is the risk if an intruder goes undetected on this system? Run `tcbck -n tree` during your idle time, and see what happens. How long did it take? Then, run `tcbck -n tree` at a peak usage time, and see what happens. Several things change the impact that `tcbck` will have on your system. These are:

- How many files are in your TCB.
- How many files you have TCB mode on for.
- How many file systems you have.
- How large your file systems are.
- Your hardware and its configuration.

If `tcbck` hurts performance too much, then give it a lower priority or nice value. One way to do this is with `nice` (for example, `nice tcbck -n tree`).

Also, you can decide that some classes of files should be checked daily, and others weekly, or monthly. However, this will not catch unauthorized set UID programs.

Here are some questions to ask. What can happen if someone goes undetected on this system for x amount of time? How long before they have valuable information? How long until they own another system, because they own this system? How does losing this machine impact your customers? What will your customers do when they find that an intruder went undetected for x amount of time?

We recommend checking as often as you can. Hourly is good. Checks can be automated, and the results mailed. We recommend automating the checks, so the program gets run even if the system administrator is on vacation or just out of the office. This is also a good reason to send the results to multiple people.

Here is a sample script that will mail root the results. If you have sendmail configured (which has serious security implications), you can mail this to anyone anywhere. Some pagers and cellar phones have e-mail capability for

instant notification no matter where you are. If you call this script tcb.sh, you can cron it to run every hour with the addition to your crontab:

```
00 * * * * /usr/local/bin/tcb.sh 2>&1 | mail -s "TCB check" root
```

```
#!/bin/ksh

TCB=/etc/security/sysck.cfg
TCB_OLD=/etc/security/sysck.cfg.OLD

/bin/cp -p ${TCB} ${TCB_OLD}
/bin/tar xf /dev/fd0 ${TCB}

/bin/tcbck -n tree
## -or- /bin/tcbck -n ALL
## -or- /bin/tcbck -n custom

/bin/diff ${TCB} ${TCB_OLD} > /dev/null ; _Rc=$?
if [[ ${_Rc} -ne 0 ]]
then
    print "Error: sysck.cfg does not match." >&2
    print "       Did you update TCB, and forget you backup?" >&2
    print "\n read/write sysck.cfg" >&2
    /bin/cp ${TCB_OLD} ${TCB}
    /bin/tcbck -n tree
## -or- /bin/tcbck -n ALL
## -or- /bin/tcbck -n custom
fi
```

6.3.2 Decide what needs to be checked

On some systems, there are so many changes that keeping those files updated in TCB would be a full time job. You can remove those files from TCB, or those attributes at least. For example, if you are adding and removing a lot of users, monitor the user files (such as /etc/passwd, /etc/security/passwd, and so on) for owner, group, mode, acl, links, and symlinks, but not for size and checksum. Otherwise, you will get multiple errors every time you add a user and then run `tcbck`.

We recommend monitoring all the AIX system files and any scripts that the root user is going to run. This will notify you if someone updates one of the root user's programs. You can either update TCB because the change was expected, or you can restore the file from backup and fix the modified files.

6.4 Understanding the report

The tcbck report can be difficult to understand. The following explains how to read output from the `tcbck -t tree` or `tcbck -t ALL` command:

3001-023 The file /dev/pts/0 has the wrong file mode.

3001-075 Change the file modes for /dev/pts/0? (yes, no) no

A *pts* is a Pseudo Terminal Slave. It will take on the ownership of whoever is logged on through that device. This will always be incorrect if someone is logged on when you run `tcbck`. Therefore, if you want to avoid getting this message, run (in ksh):

```
for i in $(ls /dev/pts/* )
do
tcbck -a ${i} mode=""
done
```

3001-041 The file /dev/rhdisk0 has too many links.

Find the links to this file, and verify them.

- If the file does not have the TCB bit set, you will have to manually find the links (with `find` or `ncheck`). Substitute your file for `/dev/rhdisk0`, and the i-node number from your `ls -i`:

  ```
  # ls -i /dev/rhdisk0
    195 /dev/rhdisk0
  # ncheck -i 195 /
  /:
  /dev/rhdisk0
  /dev/ipldevice
  ```

- If the file does have the TCB bit set, TCB will tell you the link (and prompt you to delete or add it):

  ```
  3001-032 The link from the file <new file>
           to <TCB file> should not exist.
  3001-069 Remove the file <new file>? (yes, no) no
  3001-095 Add the new link for <new file>? (yes, no) yes
  ```

  If you had answered yes to deleting the link, TCB would not have prompted you to add it.

- /dev/ipldevice is a valid file and was not added maliciously, so add it to the links section of /etc/security/sysck.cfg. If this were not the case, you would need to edit the i-node and remove the extra links.

- If /dev/rhdisk0 was linked to /hacker/rawdisk then we would be concerned and take appropriate action.

3001-089 The symbolic link from the file /usr/local/bin/rksh.test2 to /usr/bin/rksh should not exist.

If you have the TCB bit set, `tcbck -[ntpy] tree` will find unauthorized symbolic links. TCB will take the same action for symbolic links as it does for hard links.

3001-020 The file /dev/tty0 was not found.

This shows that a file that TCB is attempting to monitor no longer exists. If it was a device file, it probably is not a security problem. It may, however, be a hardware problem. In this case, the machine does not have a tty0 defined. There is one included in the sysck.cfg, so it reports the error. This can be fixed with `tcbck -d /dev/tty0` or adding the device with `mkdev`.

Chapter 7. Networks

Networks allow you to share resources. A stand-alone computer is useful, but it has its limits: every computer would need a printer, which means not only duplicated expenses, but also additional space requirements, electricity consumption, and cable handling. It is also hard to share information without a network. Sharing certain resources, including programs and data, is a natural outcome of any collaborative environment. For this reason, some say: “The network is the system.”

However, this has a drawback. On a centralized computer, ensuring centralized security is easy, and you certainly know who does what (unless a password has been stolen, of course). In a networking environment, you have a vague idea of who does what, but because the identification comes from outside, there is a risk that you can be fooled.

Today’s computing environment is more vulnerable to attacks than ever. There are several reasons for this. First, the advance of PCs in terms of performance/price ratios meant a high penetration rate in both households and organizations. Anyone can have easy access to high-performance PCs. Second, network infrastructure has improved tremendously in terms of speed and accessibility. Almost everyone is connected to the Internet in someway or another. The Internet is an unregulated and relatively anonymous network, which has both positive and negative aspects. Third, there has been a tremendous push by organizations to have a presence on the Internet, or even to push e-commerce or corporate networks (via VPN) onto the Internet. The presence of commercial information or transactions on the Internet makes it lucrative for frauds and thieves to hack on the Internet.

There are several types of hackers. Most of the press on hackers goes to those who seem to pick targets at random and try to vandalize them. Other malicious hackers target specific companies and industries to get information or extortion. Your average beginner hacker may be stopped fairly easily with a good firewall scheme. They will simply skip to the next computer that does not have a firewall. However, those that have specifically targeted your company are much more persistent. They will make use of sophisticated hacking techniques, search your phone pool for modems, and may even try to get a job at your company to find out how you do things and to insert a back door somewhere. Perhaps they already worked there and are a disgruntled employee. Either way, you must presume that they have been successful in getting past your firewall. There is also a tendency for companies to only put firewalls at Internet gateways. This completely ignores the fact that hacks can come from within the company.

Network security can be achieved in a few ways:

Network design
: Networks must be properly segregated. The worse scenario is when everything is connected into a single network with Internet access. Networks should reflect functional roles, different classes of information, as well as usage (serving clients or internal use).

Firewalls
: Network security is most often associated with firewalls. Indeed, a firewall is useful at the demarcation gateways for segregating and controlling different functional networks. The logging or audit provided at the firewall also provides a useful record of traffic flow.

Network technologies
: The increase use of untrusted networks has resulted in a demand for technologies that can make this usage more secure or at least of an acceptable risk level. Technologies include firewalls, intrusion detection systems, virtual private networks, and host network filtering.

In this chapter, we present some of the features provided by AIX that can allow networks to be designed and used in a more secure manner. The features are not mutually exclusive and may be used together or individually. These will include C2, SecureWay Directory, and IP filtering. Certain "dangerous" protocols will also be covered: NFS, NIS, and NIS+. It should be noted that the use of these protocols is inherently risky and there is a limit to how safely you can configure these protocols.

7.1 C2 security on a whole local network

The "Orange Book" criteria came from a task force of the United States Defense Science Board started in 1967. The original report, "Security Controls for Computer Systems", was published in 1970. The current document is "Trusted Computer System Evaluation (DoD85)". This material is important for many installations, but it is often misunderstood and misused.

There are two quite distinct sets of criteria. One set defines a number of security features. The other set defines the tools, information, and some of the processes required to verify the correctness (design and operation) of the features. The security features are the Security Policy and the second set is the Assurance criteria.

AIX has been C2 certified for many years now. However, up to now, this certification concerned only a stand-alone system. As soon as an AIX system

was connected to a LAN, it could not be considered C2 anymore. For more information regarding DoD classes, see Appendix A, “DoD classes” on page 205.

7.1.1 C2 network and host security

With AIX 4.3, it is now possible to have a C2 network structure on a whole LAN, or even to a set of LAN segments. Having C2 network security provides assurance that your network can be used securely within the C2 constraints. We look at some of these constrains as well as the context where the use of C2 may or may not be useful.

We be point out some possible configurations as well as constrains when using C2. It is essential that you refer to Chapter 6, “C2 Network Administration” in *IBM RS/6000 Distributed System Trusted Facility Manual (TFM)*, `http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/C2/tfm/toc.htm` for a more detailed look.

7.1.1.1 C2 network structure

The RS/6000 Distributed System running AIX 4.3 was independently evaluated and awarded a C2 rating based on The Trusted Computer System Evaluation Criteria (TCSEC). TCSEC defines a set of hierarch security ratings and established the functionality and assurance requirements for each rating. Additional information about the TCSEC is available at:

```
http://www.radium.ncsc.mil/tpep/library/tcsec/index.html
```

In summary, C2 is meant to provide controlled access protection with features such as individual accountability, discretionary access control (DAC), extensive auditing, and object reuse.

When extended to the C2 network, the LAN is taken as part of a backplane between trusted systems. It is essential that a physically secured network exists. C2 does not prevent network sniffing if there is a breach in the physical network cable. Although C2 provides features that ensure that the users cannot overreach the accesses given, there is obviously no point in having trusted systems when insufficient measures are taken to ensure that the users are trusted. There are far too many breaches that can be made within the real-world to overcome whatever protections that are placed in the computing environment. In addition, the use of r* commands require trust associations to be made. These trust associations should only be made after due diligence checks and procedures. There is strong audit and accountability provided in C2 but the point is to prevent whatever incursions from happening in the first place. There is no point having the best locks in the world if you give away the keys to the wrong people.

Figure 8 illustrates the configurations that can be used within the C2 network structure. Note that these are not the possible configurations but the *only* configurations that can be used.

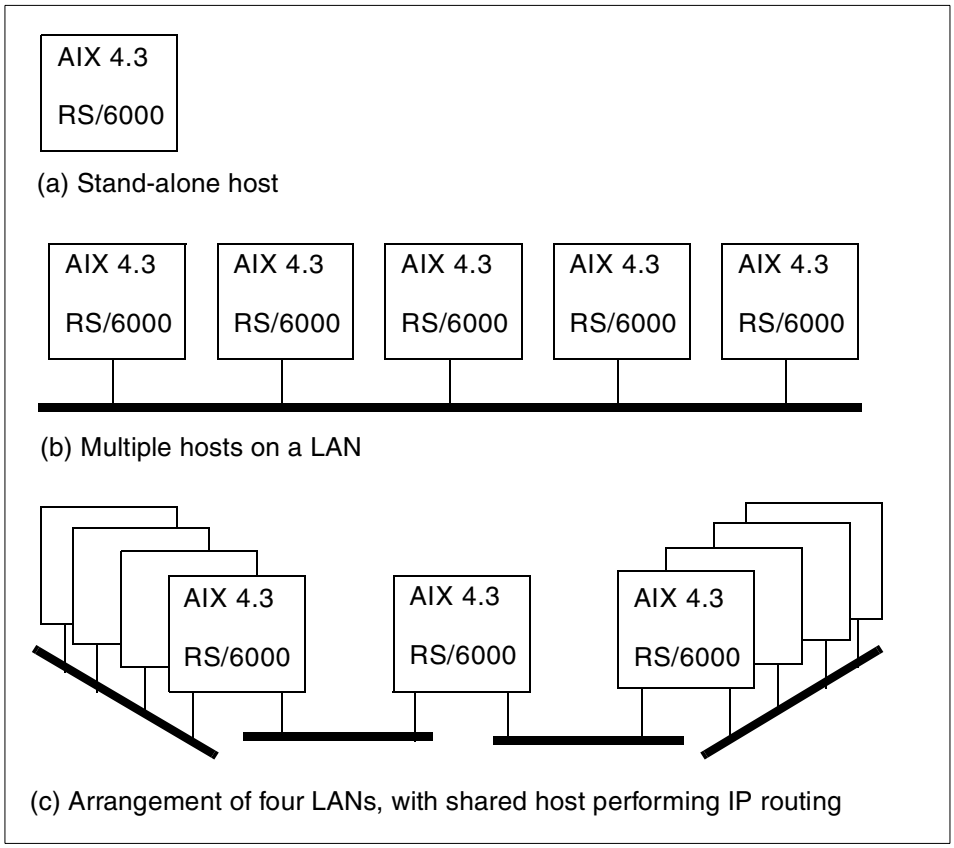

Figure 8. Configurations for C2 host and network structure

The following are the configurations shown in Figure 8:

Configuration A A single, stand-alone RS/6000 system

Configuration B Multiple RS/6000 computers on a LAN

Configuration C A system with multiple LAN segments

Besides the need for physically secure networks, routing can only be performed by the TCB, which means *no* connectivity devices, such as routers, bridges, and gateways, and definitely not to any other external networks (such as the Internet).

7.1.1.2 C2 features and constrains

This section provides a more explicit explanation of what is allowed and disallowed with a TFM, as well as security measures implemented by AIX within a C2 context.

The following are permitted network applications:

- BSD rlogin
- BSD rsh
- BSD rcp
- BSD rexec
- Telnet
- File transfer protocol (FTP)
- NFS Version 3, including rpc-stat, rpc-lock, rpc-mount, and portmapper
- BSD time synchronization protocol (timed)
- X-Windows protocol
- Simple mail transfer protocol (SMTP)
- Line Printer Daemon (LPD) protocol
- Web-based System Manager
- HTTP

No other servers are allowed to be run as root or in the kernel. Non-root servers can be run with DAC and auditing.

An important feature in AIX C2 to protect the TCP and UDP ports is known as *port protection*. Three measures are used:

- IP Security (IPSec) port filtering permits AIX to filter incoming IP packets based on combinations of source IP address (more generally, a network and netmask), protocol (TCP or UDP), and port number. This mechanism is used to disable ports that are not used in the C2 configuration.
- DACinet permits arbitrary ports (above 1024) to be designated as privileged so that they may only be bound to a socket created by the super-user. Examples would include ports used by Web-based System Manager and X11.
- DACinet also provides a means of restricting the ability of users, based on user identity, to establish connections to TCP ports. (No similar feature is provided for UDP ports.) This feature extends the IPSec address-based

notion of port filtering to permit only trusted users to establish connections to certain services (such as Web-based System Manager).

Besides port protection, AIX also provides a modified BSD r-services daemons where pre-authentication is not required.

AIX Version 4.3 System Management Guide: Communications and Networks, SC23-4127 makes mention of the following protocols, protocol architectures, protocol software, and communications hardware that is not permitted in the evaluated configuration:

- External routers, gateways, and bridges
- Modems and serial communications protocols, such as EIA 232D, V.25 bis, EIA
- 422A, X.21, and V35
- IBM legacy protocols, including systems network architecture (SNA), SDLC, HDLC, and BiSync
- Internet Protocol Version 6 (IPv6) addressing and routing
- Fiber distributed data interface (FDDI) LANs
- Internet Message Access Protocol (IMAP) and Post Office Protocol (POP)
- Basic network utilities (BNU) and UNIX-to-UNIX copy program (UUCP)
- Network Information Service (NIS)
- Dynamic Host Configuration Protocol (DHCP)
- Serial line Internet protocol (SLIP)
- Asynchronous point-to-point protocol (PPP)
- Network Install Manager (NIM)
- Network utilities, such as finger, ruptime, rwho, whois, and trpt
- Routing information protocol (RIP), along with routed daemon
- Distributed Computer Network (DCN) protocol, and gated daemon
- Distributed Computing Environment (DCE)
- Domain name protocol
- Exterior Gateway Protocol (EGP)
- Name/Finger Protocol
- Trivial file transfer protocol (TFTP)
- Distributed Computer Network (DCN) local network protocol

- Routing information protocol (RIP)
- gated, named, routed, rwhod, comsat, fingerd, talkd, tftpd, and uucpd daemons
- Simple network management protocol (SNMP) and snmpd daemon
- PC-NFS and rpc.pcnfsd daemon
- NFS remote user listing service and rpc.rusersd server
- NFS boot parameters service and rpc.bootparamd server
- NFS remote wall service and rpc.walld server
- NFS spray service and rpc.sprayd server

Note

Do not run securetcpip or install tftp, tftpd, or trpt in a C2 system. These programs were not evaluated and could bypass the security of the TCB.

7.1.2 The choice: C2 or not C2

Obviously, in a physically-secured private network with no outside connections, C2 is the best choice for security. We recommend that solution whenever possible.

However, there are stringent conditions to meet when using a C2 network structure and most environments may not be able to meet them. There is always a trade-off between security and usability. The reader has to decide on the feasibility of having C2 and its impact on the goals of the organization.

In Chapter 10, “Securing the AIX platform” on page 189, we describe how to lock down a typical AIX. Taking a slightly different track, it assumes a hostile environment and takes measures to protect itself.

7.2 SecureWay Directory

IBM SecureWay Directory is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP server.

IBM SecureWay Directory has a Web-based interface that allows the administrator to perform server and database tasks from one location, providing an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange. All that's needed is a

frame-enabled Web browser. There are no additional software requirements for the client.

LDAP is a directory listing of information about objects arranged in some order. It is suitable for storing information, such as list of books, e-mail addresses of people, and so on. A user or a program can search the directory to locate a book or an e-mail address for a specific person. It is actually a database, but the differences between general purpose databases and directories are:

- Directories are read-mostly, whereas databases tend to change rapidly.

  Because of the nature of directories that store static information, such as phone numbers, the contents do not often change. On the other hand, the information stored in databases, such as the order quantity of an item, changes rapidly.

- Data consistency requirement to directories is not strict.

  Because directories store static information, they might not have transaction support for data integrity. Duplicates and out of date data is acceptable.

IBM SecureWay Directory provides the following functions:

- Interoperability with other LDAP clients
- Secure Sockets Layer (SSL) communication
- Access control functions
- Referrals
- Server administration utilities
- Certificate management
- Client authentication
- Replication
- LDAP directory browsing through HTTP
- Simple Authentication and Security Layer (SASL)
- Client and server plug-in support
- User password encryption
- UTF-8 database support

In addition, IBM SecureWay Directory provides a Web-based administrator graphical user interface (administrator GUI) that includes online help for the administrator. From the administrator GUI, the administrator can:

- Set up the directory
- Manage day-to-day operations of the server:
  - Start up and shut down the directory server
  - Create, back up, and restore databases
- Manage access control lists
- Manage group membership
- Manage security levels (encryption options, server certificate management)
- View or change:
  - General server settings
  - General database/backend settings
  - Performance tuning options
  - Directory server activity
  - Directory replication configuration

The Directory Management Tool (DMT) also provides a GUI that enables you to manage information stored in directory servers. Use the tool to:

- Connect to one or more directory servers via SSL or non-SSL connections.
- Display server properties and rebind to the server.
- List, add, edit, and delete schema attributes and object classes.
- List, add, edit, and delete directory entries.
- Modify directory entry ACLs.
- Search the directory tree.

For information about SecureWay Directory v3.1 Administration, see:

```
http://www-4.ibm.com/software/network/directory/library/publications/31
/admin_help/parent.htm
```

For detailed product information about IBM SecureWay products, see:

```
http://www-4.ibm.com/software/network/directory/library/
```

For information about SecureWay firewalls, see:

```
http://www-4.ibm.com/software/security/firewall/
```

7.2.1 SecureWay Directory security

The IBM SecureWay Directory implementation supports SSL Version 3 for both the directory server and client. An emerging standard for World Wide Web security, SSL provides encryption of data and transport of X.509v3 public key certificates and revocation lists. The server may be configured to run with or without the SSL support. When the server is configured to support SSL (accepting connections over a secure port, defaults to 636), it still accepts connections from clients that choose not to use SSL (these clients still specify the standard unsecure port, defaults to 389). The LDAP either flows directly over TCP/IP (using the standard sockets interfaces) or over the SSL.

For server authentication, the IBM SecureWay Directory server supplies the client with the IBM SecureWay Directory server's X.509 certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure, encrypted communication channel is established between the IBM SecureWay Directory server and the client application.

For server authentication to work, the IBM SecureWay Directory server must have a private key and associated server certificate in the server's key database file.

To conduct commercial business on the Internet, you might use a widely known Certification Authority (CA), such as VeriSign, to get a “high assurance” server certificate.

IBM SecureWay Directory also supports LDAP referrals, allowing directory operations to be redirected to another LDAP directory server. Replication of the LDAP directory is supported and allows for additional copies of the directory to be available for directory read operations, which increases performance and reliability when accessing directory information.

Supported authentication
The following authentication options are supported:

- No authentication
- Simple authentication (password)
- X.509v3 public key certificate at the SSL

Kerberos authentication is not supported.

7.2.2 Configuration examples

Figure 9 shows a typical client/server network with examples of many of the capabilities provided.

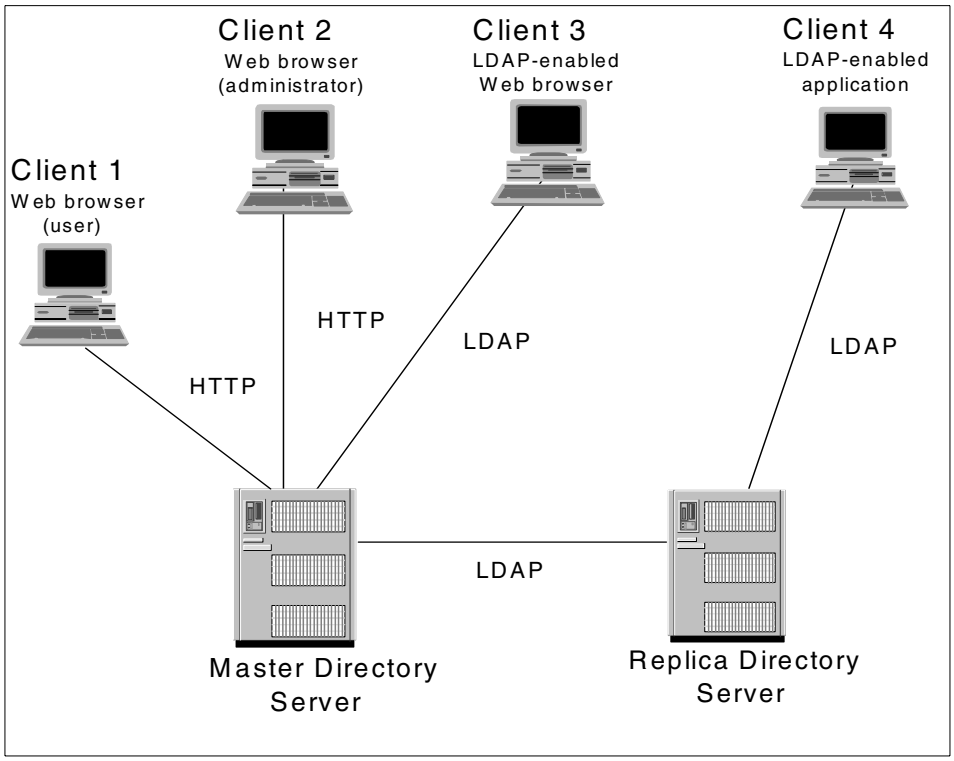

Figure 9. Typical SecureWay Directory client/server configuration

The following refer to Figure 9:

Client 1

This represents an end user of the directory working with a generic Web browser that has no built-in LDAP support. The user may connect to the server using standard HTTP and supply the URL of the HTTP gateway provided with the product. The user is then presented with a form to fill in to specify the desired search parameters. The HTTP gateway performs the search on behalf of the client and returns the matching entries from the directory to the Web browser.

Client 2

At this client, a directory administrator uses a Web browser to monitor or configure the directory. The Web browser requires no

specific LDAP capabilities. In this case, the administrator connects through a port selected during installation to a Web server located on the system containing the directory server. The HTML panels of the administration interface are presented to the administrator through the Web browser and guides them through viewing or setting configuration options for the directory.

Client 3

This represents an end user who has a Web browser that is LDAP-enabled. The protocol flow from this client to the server is LDAP; therefore, it requires no protocol conversion on the LDAP server system.

Client 4

This client is running an application that is LDAP-enabled. The application may have been built using the LDAP client toolkit provided with this product, or it may have been built using an LDAP client library from another source. Note that the client is shown connecting to the Replica server but is able to search the directory by connecting to either the Replica directory server or Master directory server.

Replica directory server

This server is shown with the IBM SecureWay LDAP Directory product installed. However, because the replication is achieved using a standard LDAP connection, the Replica server could be any LDAP server that supports Version 2 of the LDAP. Replicas are read-only. A given Master directory server may have multiple replicas configured.

Master directory server

This system runs the server software from the IBM SecureWay LDAP Directory product.

7.2.3 Installation and administration

The SecureWay Directory is a component of the base operating system for AIX 4.3; therefore, installation is done using standard installp facilities.

For detailed information on installation and configuration, we recommend the following Web site:

```
http://www-4.ibm.com/software/network/directory/library/publications/in
stall_config/aparent.htm
```

7.2.3.1 Required software

The key distinguishing feature between the IBM SecureWay server implementation and other LDAP server implementations is the use of DB2 as the backend data store.

DB2 single-user is automatically shipped with the directory server. DB2 is considered an install prerequisite on the server system. If a supported version of DB2 has been previously installed, the prerequisite requirement will be satisfied. Otherwise, the single-user version is installed with the directory server (as a separate instance of DB2).

Client

The ldap.client package contains the LDAP Client Application Development Toolkit and the LDAP Client Runtime.

Fileset names:

ldap.client.adt	LDAP Client Application Development Toolkit
ldap.client.rte	LDAP Client Runtime Environment

Requisite software must be installed before or with the filesets you are installing from this package. They are:

ldap.html.en_US.man	LDAP HTML Man Pages
bos.rte	Base Operating System Runtime

Server

The ldap.server package contains the LDAP Directory Server Administrative Interface, the LDAP Directory Server Framework, and the LDAP Directory Server Runtime Environment filesets.

Fileset names:

ldap.server.admin	LDAP Directory Server Administrative Interface
ldap.server.com	LDAP Directory Server Framework
ldap.server.rte	LDAP Directory Server Runtime Environment

Requisite software must be installed before or with the filesets you are installing from this package. They are:

db2_02_01.clp	DB2 Command Line Processor
db2_02_01.db2.misc	DB2 Utilities and Samples
db2_02_01.db2.rte	DB2 Executables
db2_02_01.client	DB2 Client Application Enabler

db2_02_01.conv	DB2 Code Page Conversions
ldap.client.rte	LDAP Client Runtime
ldap.html.en_US.admin_guide	LDAP HTML Admin Guide

Administer the SecureWay Directory server
A frame-enabled browser is required that supports:

- HTML Version 3.0 or later
- Java 1.1.8 features including JDK 1.1 AWT events
- JavaScript 1.2

The browser must be enabled to accept cookies. The following Web browsers support these specifications:

- Netscape Navigator 4.07 or higher (4.08 is recommended)
- Netscape Communicator 4.5 or higher

SSL GSKit
Global Security Kit (GSKit) is an optional software package that is required only if Secure Socket Layer (SSL) security is required. The SecureWay Directory alone does not provide the capability for SSL connections from SecureWay Directory clients. The SSL feature is added by installing the IBM GSKit package (located on the AIX Bonus Pack). The GSKit package includes SSL support and associated RSA (4) technology.

To install a secure SecureWay Directory from the SecureWay Directory package, first install the client or server from the package. Then the GSKit can be installed from the AIX Bonus Pack.

7.3 IP Security

IP Security enables secure communications over the Internet and within company networks by securing data traffic at the IP layer. This allows individual users or organizations to secure traffic for all applications, without having to make any modifications to the applications.

There are two main features provided:

Tunnels *Tunnels* provide a means of encapsulating the IP traffic between two hosts. Tunnels can be used to authenticate and/or encrypt the traffic. This is occasionally referred to as a virtual private network (VPN) in the popular press.

Filters *Filters* provide a means of providing access control on what traffic leaves or enter the host.

Figure 10 illustrates how a packet comes in the network adapter to the IP stack. The filter module filters the packet before passing it on to the tunnel definition.

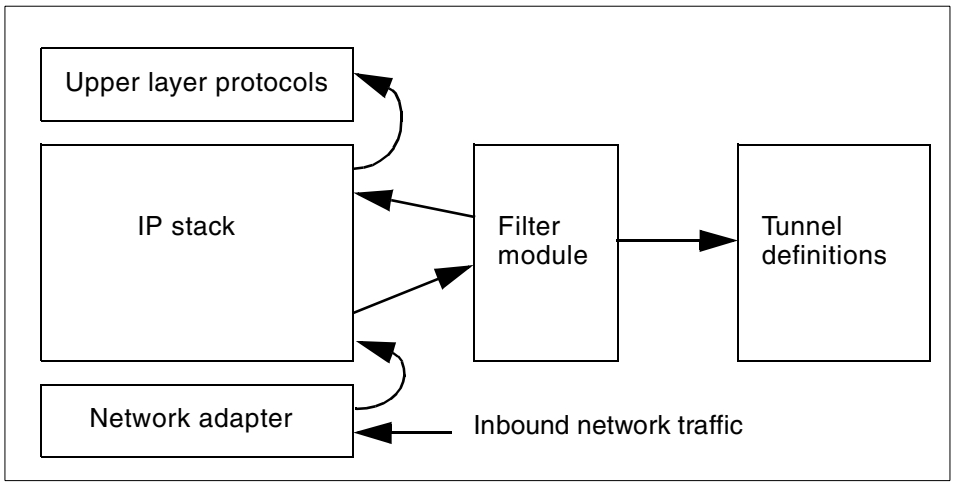

Figure 10. Relationship between tunnel and filters

Refer to *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127 for actual setup and configuration details.

The following shows the IP Security operations that are provided by SMIT. You can use fast path name ipsec4 for a quick access.

```
Start/Stop IP Security
   Start IP Security
   Stop IP Security
Basic IP Security Configuration
   Add IP Security Tunnel
   Change IP Security Tunnel
   List IP Security Tunnel
   Remove IP Security Tunnel
   Export IP Security Tunnel
   Import IP Security Tunnel
   Activate IP Security Tunnel
   Deactivate IP Security Tunnel
Advanced IP Security Configuration
   Configure IP Security Filter Rules
      List IP Security Filter Rules
```

```
   Add an IP Security Filter Rule
   Change IP Security Filter Rules
   Move IP Security Filter Rules
   Export IP Security Filter Rules
   Import IP Security Filter Rules
   Delete IP Security Filter Rules
List Active IP Security Filter Rules
Activate/Update/Deactivate IP Security Filter Rule
List Encryption Modules
Start/Stop IP Security Filter Rule Log
Start/Stop IP Security Tracing
```

Note

Web-based System Manager (WebSM) is required for Internet Key Exchange (IKE) tunnel setup.

Prerequisites

Installation and load operations are separated for the IP Security feature in AIX. The following filesets are required to install IP Security:

- bos.net.ipsec.rte
- bos.msg.LANG.net.ipsec (where LANG is the desired language, such as en_US)

And either:

- bos.crypto for CDMF support (40-bit key Commercial Data Masking Facility available on the World Trade Accessory Pack)

Or

- bos.crypto-us for DES support (56-bit Data Encryption Standard available on the U.S. Accessory Pack)
- bos.crypto-priv for Triple DES support (available in the U.S. and Canada only)

Once installed, IP Security can be separately loaded for IP Version 4 and IP Version 6. This is accomplished by issuing the `mkdev` command or through SMIT.

7.3.1 Tunnels

Tunnels encapsulate all IP traffic between the two hosts in a manner specified by the user. It provides data integrity, privacy, and authentication depending on how the tunnel is defined. Three tunnels are provided with AIX:

- Internet Key Exchange (IKE) tunnels are dynamically changing keys, established by the Internet Engineering Task Force (IETF) as a standard.
- IBM tunnels are dynamic proprietary keys.
- Manual tunnels are static, persistent keys, established by the IETF as a standard.

Figure 11 on page 139 shows a typical tunnel between two hosts.

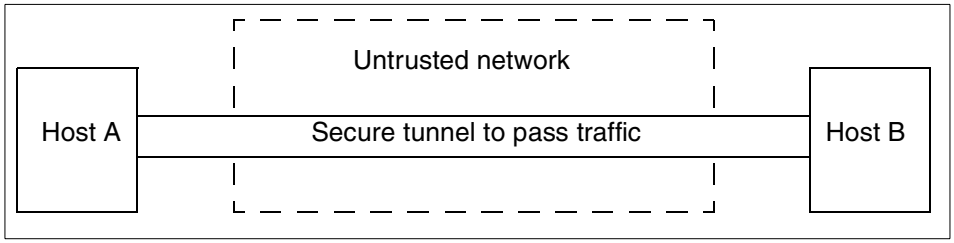

Figure 11. Secure tunnel between two hosts

IKE tunnels provide key management protocols with dynamically changing keys. This is recommended over the use of manual tunnels that use static keys. The WebSM interface is needed to configure IKE tunnels. Figure 12 on page 140 shows the WebSM window for IP Security configuration.

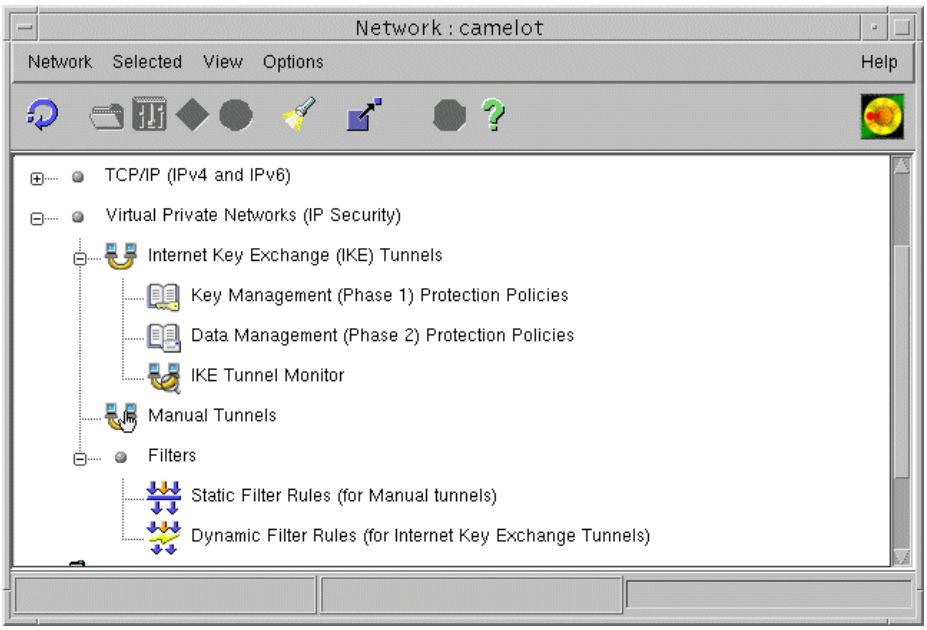

Figure 12. WebSM for IP Security configuration

A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management, SG24-5309 provides a good overview of IP Security and its configuration in various IBM products (including AIX 4.3).

There is a large number of products supporting IP Security. This includes routers, firewalls, IP Security VPN servers, and other operating systems. A combination of these components can create a secure network system. For example, your AIX workstations at your branch office may connect securely to a firewall that sits at the edge of your corporate network. We recommend that users test the IP Security interoperability between different vendor products before making any purchases.

The rest of this section concentrates on some of the decisions that have to be made when deploying IP Security.

Tunnels can be authenticated and/or encrypted. The two components that define encryption and authentication are known as Authentication Header (AH) and Encapsulating Security Payload (ESP), respectively. A new ESP allows both encrypted data and authentication digest (using Keyed MD5) in

the same packet. The different authentication and encryption schemes defined for AH and ESP vary in strength and performance.

As an example, using Triple DES CBC Encryption Module (3DES_CBC) and Hashed MAC SHA Authentication Module (HMAC_SHA) is safer than using DES CBC 4 Encryption Module (DES_CBC_4) and Hashed MAC MD5 Authentication Module (HMAC_MD5). A good reference for cryptography is *Applied Cryptography* (see C.3, “Other resources” on page 215).

How you choose a tunnel depends on what is available between both ends forming the Security Association (SA) as well as the information classification of data being transferred. Always encrypt *and* authenticate when transferring confidential information.

IKE defines how security keys are passed between the sender and receiver. There are two modes in IKE: aggressive and main. In IKE aggressive mode, only one SA is used for all communications and the identity of participants are known. Using IKE main mode provides more security by providing participant privacy. It uses two SAs: one for passing security parameters and the other for data transfer. The trade-off to higher security in main mode is the higher processing overhead.

There are also two modes for implementing SA: tunnel and transport modes. In transport mode, the original IP headers are not changed, only the payload is wrapped. In tunnel mode, the entire packet is encapsulated. Tunnel mode is used frequently between security gateways. The original destinations and source IPs are entirely hidden in tunnel mode. Tunnel mode has higher processing overhead than transport mode.

A trade-off of using IP Security, VPN, or any form of encryption is deciding on the start and endpoints of the tunnel[1]. Encryption from point to point provides security between these two points, but it also prevents the traffic from being analyzed or logged at the firewall. For example, if external parties are allowed to establish a tunnel from their machine straight to your servers to pass confidential information, they can bypass your firewalls and prevent the proxies from examining the traffic. This is a problem if the external parties cannot be trusted. One way is to have IP Security between external parties and a firewall gateway. The firewall is the tunnel endpoint. The firewall unwraps the tunnel and analyzes the traffic using application proxies before passing to the backend. The assumption is that your Internet network can be trusted because sessions after the firewall are no longer IP Security sessions. Technically, a tunnel can also be established between the firewall gateway

[1] There are two different tunnel endpoints in IP Security.

and the backend servers. These are some of the decisions that need to be made when using IP Security. See Figure 13.

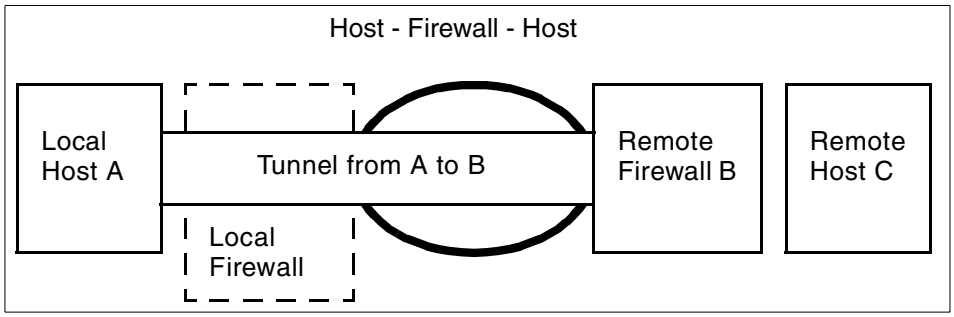

Figure 13. Host-firewall-host tunneling

Tunnels, or VPN, is a security feature. But it is necessary to understand that tunnels do not provide any additional protection on the host. Filters are provide the host protection features. If the purpose is to make use of the filters only, there is no need to create tunnels.

The actual configuration details are described in *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127.

7.3.2 Filters

Filters are excellent tools for host protection and can be used independently of tunnels. We recommend the use of filters especially in client/server architectures. An example is Internet Web servers. Only the HTTP port 80 needs to be allowed through the inbound filter or access. This closes a majority of remote service exploits. Technically, exploits may still be launched across the allowed port or via the use of denial-of-service attacks. Practically, the use of filters effectively prevents exploits from the majority of intruders.

The use of filters creates a performance overhead, and complexity increases with the number of services provided. However, the use of filters is still recommended because of the potential gains.

The manual creation of filters requires knowledge of the application protocol. A good reference book is *TCP/IP Illustrated, Volume 1: The Protocols* (see C.3, “Other resources” on page 215).

The /usr/samples/ipsec/filter.sample file contains samples using the `genfilt` command to generate filter rules. We recommend that the granularity of filters be changed to meet your specific needs.

There are a few points to note about filters:

- Granularity of filter rule definition is important. Filter fields need to be specifically defined for the protocol rather than using catch-all values, such as “all” or “0.0.0.0”.
- Automatically-generated rules permit all traffic over the tunnel. User-defined rules can place restrictions on certain types of traffic. These user-defined rules should be placed before the automatically-generated rules. Remove automatically-generated rules if they are not needed.
- Order. Rules are applied down the list. Order them by the number of times they are accessed. The most commonly used ones should be defined first.
- Return packets. In TCP, return packets are defined with ack bits. This is important to prevent a handshake access on the port while spoofing as returning traffic.
- Deny-all. Insert a deny-all rule into the filter if it is not present. The following shows rule 100 as a deny-all rule:

  ```
  100 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no all any 0 any 0 both both
  yes all packets
  ```

- Logs. Deny logs are useful for detecting potential intrusions. Permit logs are useful as a means of measuring usage and audit purposes.
- False-positives. This occurs when you use the deny logs as a means of detecting intrusions but there is too much “noise” in the traffic to prevent effective detection. There are two ways to solve this problem. Either use a log management software or insert deny rules that do *not* log the false-positives. There may be a danger in doing the latter, especially if the intruder knows it and hides their traffic with the false-positive rule.
- Residual risks. The use of filters alone is not sufficient if the intention is to protect confidential data. Filters and tunnels can be used together to achieve this purpose
- Fixes. It bears repeating that the latest fix needs to be applied since this may fix some IP filtering bugs.

Example of using IP filters on a Telnet host

The best way to start using filters is to try it on a test machine. Using the `smit ips4_advanced` fast path, it is possible to access a range of filter capabilities. This example illustrates the use of filters with an example for Telnet from 9.12.0.129 to 9.12.0.19.

Figure 14 shows how a server with filters can allow only selective access to itself (9.12.0.19). In this case, only Telnet access from 9.12.0.129 is allowed.

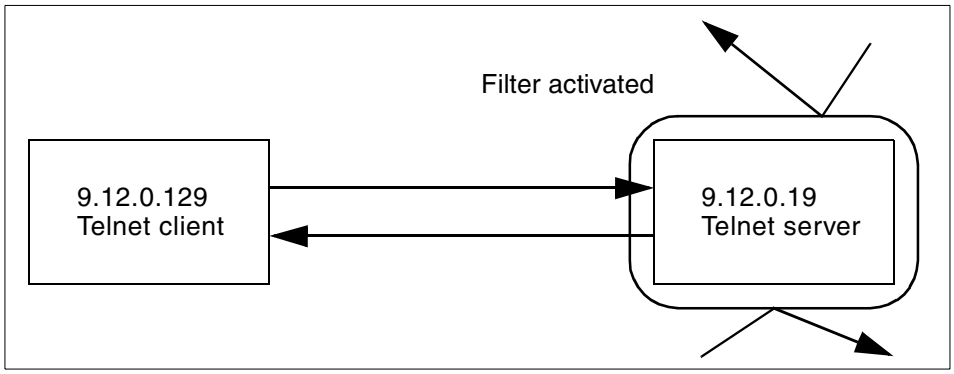

Figure 14. Using IP filters

To start, you can use SMIT to access the Advanced IP Security settings. Use the `smit ips4_advanced` fast path. The following shows the SMIT menu:

```
Configure IP Security Filter Rules
List Active IP Security Filter Rules
Activate/Update/Deactivate IP Security Filter Rule
List Encryption Modules
Start/Stop IP Security Filter Rule Log
Start/Stop IP Security Tracing
```

To start adding filter rules, select **Configure IP Security Filter Rules** from the SMIT menu. Alternatively, use the `smit ips4_conf_filter` fast path. The following shows the SMIT menu:

```
List IP Security Filter Rules
Add an IP Security Filter Rule
Change IP Security Filter Rules
Move IP Security Filter Rules
Export IP Security Filter Rules
Import IP Security Filter Rules
Delete IP Security Filter Rules
```

To add a filter rule, select **Add an IP Security Filter Rule** from the SMIT menu. Alternatively, use the `smit ips4_add_filter` fast path. The following shows the SMIT menu:

```
 Add an IP Security Filter Rule

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Rule Action                                      [permit]
* IP Source Address                                []
* IP Source Mask                                   []
  IP Destination Address                           []
  IP Destination Mask                              []
* Apply to Source Routing? (PERMIT/inbound only)   [yes]
* Protocol                                         [all]
* Source Port / ICMP Type Operation                [any]
* Source Port Number / ICMP Type                   [0]
* Destination Port / ICMP Code Operation           [any]
* Destination Port Number / ICMP Type              [0]
* Routing                                          [both]
* Direction                                        [both]
* Log Control                                      [no]
* Fragmentation Control                            [all packets]
* Tunnel ID                                        [0]
* Interface                                        []
```

First, you define the Telnet establishment packets from 9.12.0.129 (Telnet client) to 9.12.0.19 (Telnet server).

```
 Add an IP Security Filter Rule

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Rule Action                                      [permit]
* IP Source Address                                [9.12.0.129]
* IP Source Mask                                   [255.255.255.255]
  IP Destination Address                           [9.12.0.19]
  IP Destination Mask                              [255.255.255.255]
* Apply to Source Routing? (PERMIT/inbound only)   [no]
* Protocol                                         [tcp]
* Source Port / ICMP Type Operation                [gt]
* Source Port Number / ICMP Type                   [1023]
* Destination Port / ICMP Code Operation           [eq]
* Destination Port Number / ICMP Type              [23]
* Routing                                          [local]
* Direction                                        [inbound]
* Log Control                                      [yes]
* Fragmentation Control                            [all packets]
* Tunnel ID                                        [0]
* Interface                                        [tr0]
```

Next, you define the return packets from 9.12.0.19 (Telnet server) to 9.12.0.129 (Telnet client).

```
 Add an IP Security Filter Rule

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Rule Action                                        [permit]
* IP Source Address                                  [9.12.0.19]
* IP Source Mask                                     [255.255.255.255]
  IP Destination Address                             [9.12.0.129]
  IP Destination Mask                                [255.255.255.255]
* Apply to Source Routing? (PERMIT/inbound only)     [no]
* Protocol                                           [tcp/ack]
* Source Port / ICMP Type Operation                  [eq]
* Source Port Number / ICMP Type                     [23]
* Destination Port / ICMP Code Operation             [gt]
* Destination Port Number / ICMP Type                [1023]
* Routing                                            [local]
* Direction                                          [outbound]
* Log Control                                        [yes]
* Fragmentation Control                              [all packets]
* Tunnel ID                                          [0]
* Interface                                          [tr0]
```

To list the filter rules, select **List IP Security Filter Rules** from the Configure IP Security Filter Rules SMIT menu. You see output similar to the following:

```
1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both
both no all packets 0 all
2 *** Dynamic filter placement rule for IKE tunnels *** no
3 permit 9.12.0.129 255.255.255.255 9.12.0.19 255.255.255.255 no tcp gt
1023 eq 23 local inbound yes all packets 0 tr0
4 permit 9.12.0.19 255.255.255.255 9.12.0.129 255.255.255.255 no tcp/ack
eq 23 gt 1023 local outbound yes all packets 0 tr0
0 deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no
all packets 0 all
```

The output tells you the following information:

- Rule 1 is placed automatically.
- Rule 2 is a comment (*).
- Rule 3 allows an inbound Telnet from host 9.12.0.129 to 9.12.0.19.
- Rule 4 allows the outbound Telnet return packet from host 9.12.0.19 to 9.12.0.129.

- Rule 0 is a deny-all rule dynamically generated when **Deny All Non_Secure IP Packets** is chosen during Start IP Security.

After making sure that the rules are correct, start IP Security by selecting **Start/Stop IP Security** from the following SMIT menu. You can access the following SMIT menu by using the `smit ipsec4` fast path:

```
Start/Stop IP Security
Basic IP Security Configuration
Advanced IP Security Configuration
```

While using SMIT to add rules is helpful, if you want to use a similar set of rules for the other machine or to add a lot of rules, it might be efficient to use a text editor. In this case, you can import and export rules.

To export filter rules, select **Export IP Security Filter Rules** from the Configure IP Security Filter Rules SMIT menu. Alternatively, use the `smit ips4_export_filter` fast path. You see the following SMIT menu:

```
 Export IP Security Filter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Export Directory Name                              [/tmp]
* Filter Rules                                       []

  +--------------------------------------------------------------------------+
  |                             Filter Rules                                 |
  |                                                                          |
  | Move cursor to desired item and press Esc+7. Use arrow keys to scroll.   |
  |     ONE OR MORE items can be selected.                                   |
  | Press Enter AFTER making all selections.                                 |
  |                                                                          |
  |   1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both b |
  |   2 *** Dynamic filter placement rule for IKE tunnels *** no             |
  |   3 permit 9.12.0.129 255.255.255.255 9.12.0.19 255.255.255.255 no tcp g |
  |   4 permit 9.12.0.19 255.255.255.255 9.12.0.129 255.255.255.255 no tcp/a |
  | 0 deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both n   |
  |                                                                          |
  | F1=Help                F2=Refresh             F3=Cancel                  |
  | Esc+7=Select           Esc+8=Image            Esc+0=Exit                 |
  | Enter=Do               /=Find                 n=Find Next                |
  +--------------------------------------------------------------------------+
```

An export file is created in the directory you have chosen. In this example, it is /tmp/ipsec_fltr_rule.exp. You have the capability to choose the rules for export. The following example chose rules 3 and 4 to be exported.

```
 COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

Filter rule 3 for IPv4 has been exported successfully.
Filter rule 4 for IPv4 has been exported successfully.
Filter rule(s) have been exported to /tmp/ipsec_fltr_rule.exp successfully.
```

After editing the exported filter rules file with a text editor, you can import the file. To import the filter rules file, select **Import IP Security Filter Rules** from the Configure IP Security Filter Rules SMIT menu. Alternatively, use the `smit ips4_import_filter` fast path. The file name must be ipsec_fltr_rule.exp.

7.4 TCP/IP base services

This section describes the security of TCP/IP services and provides an example of the things to look for when configuring these services. This is not meant to be a comprehensive guide but more of how a user can approach the security of numerous services that are present. Note that the services mentioned in this section can be considered dangerous (not withstanding the security measures recommended). Do *not* use them unless necessary.

For more information on TCP/IP, refer to the following documentations:

- Chapter 3, “Transmission Control Protocol/Internet Protocol” in *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126.
- Chapter 3, “Transmission Control Protocol/Internet Protocol (TCP/IP) Overview” in *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127.
- *AIX Version 4.3 Commands Reference*, SBOF-1877.

There is no doubt that TCP/IP services are required. The more important point when running these services is to ask the right questions. With this in mind, the following is a short FAQ.

Which services are required?
There is never a need to run more than necessary. This point has been mentioned in several other chapters but is worth repeating. So, the answer is to only run whatever service is required and not start the others (or remove them). Each additional service is a potential vulnerability.

How can they be configured to be more secure?
All services have their own manner in authentication and access. Therefore, turn on authentication where possible. Do not depend on trust relationships. And, if access can be controlled in terms of virtual directories or permissions on the server side, do it! Always read the man pages or manual for configurable options for the daemons you are running.

What can be done if authentication is not secure?
This has become a huge problem as TCP/IP services emerge from trusted environments to hostile networks, such as the Internet. Most services, such as Telnet, FTP, SNMP v1, POP3, and HTTP, do not have strong authentication and passwords travel across the networks in plain text. In this case, the risk of running such a service has to be taken into account. If it is a public service, there is probably more restrictions on what you can do. The use of HTTPs, ssh, and IPSec are some of the ways that can secure the session. But invariably, they will involve some changes at the client application. HTTPs is probably the only widespread encryption protocol due to high penetration of Web browsers on the desktop.

7.4.1 TFTP

TFTP is a protocol simpler than FTP, requiring no password (it is run by user *nobody*). It does not allow listing directories. It is sometimes used to boot workstations or terminal servers. Network management stations may also use TFTP for transferring router configurations. There is absolutely no reason to use TFTP unless necessary.

There are a few measures that can be made to limit the risk of running tftp.

Access control—The /etc/tftpaccess.ctl file is searched for lines that start with allow: or deny:. Other lines are ignored. If the file doesn't exist, access is allowed. The allowed directories and files can be accessed and the denied directories cannot be accessed. Further information and example configurations for Xstations, diskless clients, and restricted entry can be found in the /usr/samples/tcpip/tftpaccess.ctl file.

Turning on the tfptd options listed in Table 13 will allow better logging and control.

Table 13. tftpd options

option	description
-d Directory	Specifies default destination directory. The directory specified will be used as the home directory for storing files only. This default directory will be used only if a full path name is not specified. The default directory for retrieving files is still /tftpboot.
-i	Logs the IP address of the calling machine with error messages.
-v	Logs information messages when any file is successfully transferred by the tftpd daemon. This logging keeps track of who is remotely transferring files to and from the system with the tftpd daemon.

7.4.2 Telnet or rlogin?

The answer is neither. Both the telnet and rlogin commands are unsecure means of communicating if it takes place across untrusted networks. Instead, make use of ssh or IPSec (available on AIX) or other forms of virtual private networks (VPNs) wherever possible.

For historical reasons, similar UNIX commands sometimes come in two flavors. This is the case for remote access and file transfer commands: there are `telnet` and `ftp` on one side and `rlogin`, `rcp`, and `rsh` on the other. Table 14 summarizes the differences.

Table 14. Comparison of some remote access schemes

Function or value	telnet and ftp	rlogin and other r commands
Open a terminal session on a remote host	User supplies password	rlogin daemon on the accessed computer checks in its files if the accessing computer is trusted. If not, user supplies password.
Allow or forbid access by other computers	none	System-wide: /etc/hosts.equiv Specific to a user: $HOME/.rhosts
Companion commands using the same security scheme	ftp	rcp (remote copy), rsh (remote shell)

Function or value	telnet and ftp	rlogin and other r commands
User efficiency	Comparable to any other terminal emulation or file transfer program	Enable more efficient work on multiple machines. No password to retype, no context changes.

There are also two schools of thought here. The minority says that rlogin, rcp, and rsh are, in a way, more secure (or less insecure) than telnet and ftp, because neither of the r commands transmits any password on the network, thus this password cannot be eavesdropped. They are probably right for small networks. The majority objects with the following reasons:

- The $HOME/.rhosts files contain the names of hosts that a user trusts. They are an open door for network penetration should an account be compromised.
- The /etc/hosts.equiv file presents a similar menace for all user accounts.

Consider also that the .rhosts file allow any user to grant a password-free remote access on his/her account to any external computer without asking or even telling the system administrator; as such things are not considered good for security, nor for the ego of the system administrators, r commands are often removed by system administrators.

Experience also shows that large networks have been penetrated in the past mainly because of hackers gaining access to these security files.

However, the argument is pointless since both r commands, ftp, and telnet are just not the right applications to use when the network is untrusted.

7.4.3 The securetcpip command

To get rid once and for all of untrusted `tftp` commands, r commands and the associated daemons, AIX has a `securetcpip` command. The `securetcpip` command provides enhanced security for the network.

Note that this command makes sense only for non-C2 networks, since the r commands are not supposed to present any risk in C2 networks. See 7.1, “C2 security on a whole local network” on page 124. Of course, the people and physical infrastructure are assumed to be trusted.

When the `securetcpip` command is executed it runs the `tcbck -a` command that disables the untrusted commands and daemons:

- rcp

- rlogin
- rlogind
- rsh
- rshd
- tftp

The disabled commands and daemons are not deleted; instead, they are changed to file mode 0000. You can enable a particular command or daemon by re-establishing a valid mode.

An easier way to look at these daemons is simply not install the server fileset in the first place. This is because many of the other services are not secure anyway. For example, the SNMP server allows a local client to have read access—not something that should be allowed in most environments.

7.5 Network file system (NFS)

The object of NFS is to share file systems on a host with external hosts. This section discusses the security issues associated with NFS.

For more information on NFS, refer to Chapter 10, “Network File System” in *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127.

7.5.1 NFS principles

The network file system (NFS) provides a method of accessing files and directories on other machines on the network and treating them as if they were local. A directory on a remote machine can be mounted onto the local file system. It then appears to be local to the users. All actions performed on this mounted directory are automatically passed across the network to the remote host for processing.

The three most common uses of NFS are:

- Sharing data files of given directories between computers.
- Using disk space on an alternate computer.
- Using a direct-access device (for instance a CD-ROM reader) available on another computer.

A host sharing its local file systems with other hosts is called a *server*. Proposing these file systems for sharing is called *exporting*. The file systems

that may be exported and the permissions or restrictions associated with them are listed in the server's /etc/exports file.

A host mounting a remote file system is called a *client*. The client system must mount the exported file system on a local *mount point* before it can use it. The mount point can be any directory (preferably empty)[2].

On a client system, issuing the `mount` command shows the mounted file systems: remote file systems are preceded by their original host name, followed by a colon (:). On a server system, you can see which remote hosts are using its exported file systems by using the `showmount` command.

Mounting a remote file system is not necessarily automatic, although placing the relevant stanzas in /etc/filesystems can solve this. Automatic mounting should be avoided unless user access is carefully controlled. Consider using shell scripts or commands to mount remote file systems only when needed. This can prevent file systems from being mounted unnecessarily and thus somewhat reduce security exposures. The use of *NFS mount groups* makes this mounting on demand very easy when many file systems have to be mounted or unmounted together.

For an NFS environment to have any security, the administrators on all server and client systems must be trustworthy. If workstation owners are their own administrators, all the workstation owners involved must be trustworthy (in the sense of trustworthy security administrators).

7.5.2 The /etc/exports file

The list of directories that are exported from a server is maintained in the /etc/exports file. Each directory must have a separate entry. No one should directly edit this file. Use `smit nfs` instead. A typical entry might be:

```
/usr/custdata -access=clienta,ro
```

This allows the directory /usr/custdata to be exported only to the system "clienta". It is exported read-only so that changes cannot be made by the remote system.

It is important to restrict access to any exported directory. The default, with no options specified, is to export a directory to all users with read-write permissions. A variety of controls are available within the /etc/exports file, including control of root access via NFS.

As an administrator, you have two particular concerns with /etc/exports files:

[2] If the directory used as a mount point contains local files, these local files will be inaccessible while the remote mount is in effect.

- For your “official” NFS servers, do the /etc/exports files provide sufficient protection? Are only needed directories exported? Could deeper-level directories be used instead?
- For other hosts, do /etc/exports files exist? If so, they imply that system may be functioning as an NFS server. Does this comply with your security policies?

When the `exportfs -a` command runs, the directories listed in /etc/exports become available (/etc/rc.nfs shell script runs `exportfs -a` at startup time). Run `exportfs -a` again to validate any changes made to /etc/exports file.

The `exportfs` command interrogates /etc/exports and updates accordingly the /etc/xtab file, which holds the list of currently exported directories. Do *not* edit /etc/xtab directly! Use `exportfs -a` to update /etc/xtab.

Resist the temptation to export more than what is strictly needed just in order to avoid typing paths. For instance, if you wish to export /usr/lpp/appl1/bin and /usr/lpp/appl2/bin, you should export two directories (have two entries in /etc/exports):

```
/usr/lpp/appl1/bin
/usr/lpp/appl2/bin
```

Rather than the shorter directory entry:

```
/usr/lpp
```

7.5.3 NFS support for access control lists

NFS under AIX supports the AIX file system ACL model: a remote procedure call (RPC) layer program that passes ACL information between the client and server.

This may lead to some unexpected results. ACL support is an additional function that has been added to AIX and, as such, it does not change the NFS protocol specification. Non-AIX NFS clients may not even *see* the ACLs and, in that case, will not respect them. Such problems will not arise if the client is also an AIX system.

ACL functions will not work in heterogeneous networks.

See 5.6, “Access control list (ACL)” on page 100 for more information about ACLs within AIX.

7.5.4 Secure option

NFS provides a secure option. Properly used, this option provides secure authentication of NFS *users*. It does not provide additional data protection, encryption of LAN traffic, or encryption of data. It does provide protection against counterfeit host connections (that is, a system that claims another systems name and/or IP address). In practice, the secure option is not often used.

To use the secure option, you should:

- Install and use a DES authentication method (NIS+ and DCE provide DES authentication).
- Have a basic understanding of public key cryptography.
- Learn to use `keylogin` when you encounter login problems.
- Install time daemons in every system.

Do not use the secure option without studying it well. Build and use a trial environment before using it in a larger production environment.

Setting up the secure option

To mount a directory with secure option, both ends of the connection must be set up correctly. On the server, the /etc/exports file must have an entry for the directory to be exported with the secure option set as follows:

```
/usr/secretdata -secure
```

Where `/usr/secretdata` is the exported directory. Other restrictions on the directory may also be specified.

The directory should be exported using the `exportfs -a` command. It is worth checking that the /etc/xtab file also shows that the directory is to be exported with the secure option.

On the client, there must be an entry in /etc/filesystems for the securely mounted directory. This entry is of the form:

```
/usr/secretdata:
dev = /usr/secretdata
nodename = rs6000
vfs = jfs
mount = true
check = true
options = secure
```

Other options, such as `ro`, can be specified as well. If the secure option is not specified, DES authentication will not be used, and the directory will be exported using the standard AIX authentication scheme.

If you wish to use this secure mount, you must configure DES authentication on your system first. DES authentication security:

- Encrypts the current time of each client request.
- Compares this with the encrypted system time on the server.

This prevents a request to be sniffed on the network and used at a later time. A discrepancy of one minute is allowed to account for network delays.

If timed [3] is running, and there is a synchronized network time, this is automatic and transparent. If not, the client asks the server for its time and then calculates by how much the client system time differs and does an appropriate time conversion.

Secure NFS also uses a public key system with public and private (secret) keys. The public key system is used to establish initial DES keys for a session. The public keys are established with the `chkey` command or by the system administrator with the `newkey` command. This prompts for the user's password and then generates an encrypted key-pair containing a public key and secret key.

The client and server must share the same encryption key to begin a conversation. The client generates a random conversation key used to encrypt the timestamp. This is calculated from public and secret keys in such a way that the server does not need to know the client's secret key and vice versa. Only the client and server can calculate this because doing so requires knowing one secret key or the other. An outside agent cannot calculate the secret key by any reasonable means.

The conversation key is encrypted using a public key and sent within a packet known as a credential[4]. This is the only time a public key is used for encryption.

The credential contains the name of the client, the encrypted conversation key, and a variable window that is itself encrypted by the conversation key. The window contains the encrypted timestamp and an encrypted verifier for the window. This makes it much harder to guess the credential. The server stores this information in a credential table and replies to the client. In the returned verifier, the server sends an index ID to the credential table plus the

[3] timed is a TCP/IP-related daemon that synchronizes the system time across multiple systems.
[4] Note that base AIX, Secure NFS, and NIS+ each have their own separate definition of what are credentials.

client timestamp minus one, again encrypted by the conversation key. Because only the client knows the original timestamp that was sent within the credential, only the client will know the value of the timestamp minus one. Therefore, the client can verify the server. This occurs whenever service is requested by the client system.

Note that the encryption key is sent back and forth only at the start of the conversation. It is not sent on each client transmission. This makes it much harder to determine the key and “break the code.”

This whole process is only to verify the identity of the client and server. User data is *not* encrypted.

7.5.4.1 The client-server DES interaction

The Secure NFS service relies on the DES encryption facility, which, in turn, requires public and private keys. These keys are usually generated for the user from that user’s login password. To remain synchronized, the password is updated in a central point in the network maintained by NIS+ and the `passwd` command (formerly `yppasswd` with the old NIS).

7.5.5 NFS security risks

Let’s suppose that user 101 is bob on the accessing computer and alice on the accessed computer. With regular (non-secure) NFS, bob would be able to access all the NFS-mounted file systems accessible to alice without even having to know her password.

For this reason, the three following rules are generally observed when using standard NFS:

- NFS-exported file systems should be advertised system wide as world-open and no sensitive data should be stored in them.
- Mounted file systems should be considered potentially unsecure and not allowed to run under root control or have SUID bit honored (nosuid).
- NFS exports should be controlled as much as possible. See Table 15.

Table 15. Controlled NFS access

Type of data	Remote read/execute access	Remote write access
Unclassified data or programs (for example, compiled sharewares)	Open (it is not necessary to specify a host list)	Preferably none

Type of data	Remote read/execute access	Remote write access
Important programs or data	Specify an explicit host list	Specify an explicit host list

7.5.5.1 Restricting plain NFS mounts

Here is the SMIT screen to add a directory to the export list:

```
 Add a Directory to Exports List

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields]
* PATHNAME of directory to export                 []                        /
* MODE to export directory                        read-write                +
  HOSTS & NETGROUPS allowed client access         []
  Anonymous UID                                   [-2]
  HOSTS allowed root access                       []
  HOSTNAME list. If exported read-mostly          []
  Use SECURE option?                              no                        +
  Public filesystem?                              no                        +
* EXPORT directory now, system restart or both    both                      +
  PATHNAME of alternate Exports file              []
```

MODE to export directory

It is a good practice to have separate exported directories for read-write access and read-only access, even to the same file system (*read-mostly* is often preferred to *read-write* for performance reasons, but this has no impact on security).

HOSTS & NETGROUPS allowed client access

If you do not specify any hosts here, then *all* hosts (and netgroups) have client access[5].

Anonymous UID

This specifies the UID that will be assigned, among others, to client root users when accessing files from this exported directory.

HOSTS allowed root access

Preferably, no host should have remote root access, because otherwise, this host could access all the files in the mounted file system, a possible security threat. Nevertheless, if you want some machines to have root access to this directory, specify here the list of those you are administrating.

[5] A logical choice. Why would you export the file system if you wanted nobody to use it?

7.6 Network information system (NIS)

Separately accessing half a dozen of computers is tedious for both the user and the system administrator. The administrator has to create each new user on each machine (see Chapter 4, “Passwords” on page 57). The users must change their password on each machine and remember their passwords for all the machines. This is especially difficult if the user has not accessed a machine for some time.

Mounting the /etc/passwd of one machine on all other machines might be a solution. Indeed, this is precisely what is done on an AIX C2 network.

If the network is not C2, this is unacceptable since passwords would be available to anyone via NFS. Therefore, you must select something more complex, such as NIS, NIS+, DCE, and so on.

7.6.1 NIS principles

A solution is to store user information in some other structures (called *maps*) and have daemons synchronizing the maps between the different computers. Entries in /etc/passwd file will still exist, but only for users that have to be unique to a machine; *root* is an example of such a user. The computer responsible for storing the reference map will be called the NIS server. The computers using these maps (or using local copies maintained by the NIS daemons) will be called the NIS clients.

7.6.2 Why NIS is used

NIS is still used because when it is in operation:

- User IDs are coherent throughout the whole computer set.
- A user password can be changed from any machine, and the change will be reflected to other machines automatically (with a slight delay).
- Some users, for instance university trainees, can still be defined locally to a machine (root *must* be defined this way). In such case, it is good practice to assign them a given user number range (for example, over 10000) to distinguish them at a glance.

7.6.3 NIS security risks

NIS brings some security risks as follows:

- Whenever users change their passwords, these passwords are transmitted on the LAN in a non-encrypted form.

- Whenever a new host issues an ypbind (that is, asks to be an NIS client), all NIS maps are transmitted to the host in a non-encrypted form.
- Anybody knowing or guessing the NIS domain name can get these NIS maps, including the password map, by asking to become an NIS client without the system administrator being notified (the list of allowed clients is not specified on the server).

Worst of all: not only are all sensible information (maps) in regular files, but some freeware programs available on the Internet (such as *pscan*) will attempt to dump these maps to an attacking site.

NIS is generally used on small networks not connected to the outside world. Now that NIS+ is available with AIX 4.3, NIS should not be used anymore. While using an NIS compatibility mode on NIS+ is possible, it is better from both a security and ease of administration point of view to migrate fully to NIS+ *without* installing any NIS compatibility.

7.7 NIS Plus information system (NIS+)

Despite its name, NIS+ is not a modification of NIS. It is a completely different product, using different files, which are also shipped with AIX 4.3[6]. Directly installing NIS+ is less cumbersome than migrating from NIS to NIS+.

We had reports from many system administrators that enabling the NIS compatibility mode on NIS+ is not a good idea, leading to both extra system administration time and to security risks.

For more information, refer to the following documents:

- Chapter 6, “Network Information Service (NIS)” in *AIX Version 4.3 Technical Reference: Communications Volume 1*, SC23-4161.
- *AIX Version 4.3 Network Information Services (NIS and NIS+) Guide*, SC23-4310.

7.7.1 NIS+ and security

From a security point of view, the main interest of NIS+ is that it provides user authentication through the data encryption standard (DES) as a default choice. This way, all NIS+ administrative information, including passwords, is encrypted before going onto the network. The passwords, already encrypted in /etc/security/passwd (but, in a way, that could still be guessed by cracker

[6] If you install NIS+ from the bos.net.nisplus fileset, the filesets bos.net.nis.server and bos.net.nis.client will, however, be automatically installed as prerequisites because a part of them is required for NIS+ to work.

programs available on the Internet), get an additional level of encryption that way.

NIS+ requires you to specify NIS+ clients. This is a feature that NIS lacked. NIS+ ensures that only authorized computers have access to your NIS+ domain.

Additional features in NIS+ concern easier or more powerful system administration rather than security. For example, it is possible to have a whole hierarchy of domain names rather than a flat space like the NIS.

With NIS+, time matters

NIS+ asks for 17 intermediate security operations to be performed every time two machines have to exchange NIS+ information. Therefore, expect these operations to be slower.

NIS+ operations will fail if the time difference between two machines is greater than 60 seconds. For that reason, machines must use a time server to synchronize their clocks; otherwise, NIS+ operations may repeatedly fail.

To install a time server, see ntp, xntp, timed, and setclock in *AIX Version 4.3 Commands Reference*, SBOF-1877.

Improperly configured, simple NIS+ operations can also take up to 40 seconds to be competed, which means that two operations out of three may fail. If you do not use IPv6, be sure to use bind4 instead of bind in your /etc/netsvc.conf file.

7.8 Domain Name System (DNS) and IP addressing hacks

DNS refers to a hierarchy of names separated by dots that are mapped to IP addresses (and vice-versa). Basically, it is a host name to IP lookup protocol. On UNIX systems, including AIX, this is managed by BIND (the Berkeley Internet Name Daemon). DNS and BIND are so closely related that it is not uncommon to see one of these used to mean the other. For example, “which version of DNS are you using?” refers, properly speaking, to the BIND version used.

We refer here to bind8 (IPv6), which allows dynamic updates. The faster and more classic bind4 (IPv4), which is still shipped, is not concerned with the extensions.

7.8.1 DNS principles

When asked to resolve a host name, for example, sales.west.france.acme.com, the name server tries to answer the following three questions:

1. Do I have the answer in my DNS cache?
2. If not, does this name belong to my domain?
3. Otherwise, to which other name server should I escalate this request?

Escalating the request and getting back the answer can be quite time consuming. If your name server domain is on a completely different branch, for example, abc.xyz.fr, the following domain name servers will be queried in sequence (in the worst case, that is with no cache hit anywhere):

1. com[7]
2. acme.com
3. france.acme.com
4. west.france.acme.com

To reduce both network load and delays, each name server manages a cache storing the answers to its most recent queries for some time. Once your name server finally gets the IP address for west.france.acme.com, it will not have to make the four preceding queries anymore. In the same way, you probably use your personal phone book as a very small cache for a subset of all the public and corporate phone books you deal with in the world.

The fact that name servers use caches allows to speed name resolving by one to three orders of magnitude (a typical resolution time is 30 ms, although some replies may take many seconds), but raises the question of *cache updating*, since the Internet contains millions of hosts, thousands of addresses change every day. If an address associated with a name changes, and the caches referring it are not updated, the non-updated caches will forbid normal name resolution instead of speeding it. We come back to that point in a while.

7.8.2 DNS spoofing

According to dictionary.com, spoofing means: nonsense; a hoax; a gentle satirical imitation; a light parody.

A host only knows its name server by its IP address as defined in /etc/resolv.conf. This IP address can be faked by an attacking machine on the

[7] We assume that at least the root name servers are defined locally in order to bypass lookups in xyz.fr and fr.

LAN (for instance a portable computer). If that false name server answers faster than the real name server[8], it can misdirect the querying users to other, hostile, computers. However, this technique has three drawbacks from a hacker's point of view:

- It requires an available machine on the LAN.
- Success is not guaranteed—even if it is faster, the false name server will only fool the hosts querying its IP address for the first time (that is, just booting); other hosts already have the real name server MAC address in their cache.
- The same IP address associated with different MAC addresses can be noticed by a system or network administrator, especially since it is may cause unintended trouble.

Therefore, hackers soon designed the following scheme, known as DNS spoofing.

7.8.2.1 DNS spoofing principle

We saw in 7.8.1, "DNS principles" on page 162 that cached addresses need to be updated whenever they change. The whole idea behind DNS spoofing is to send counterfeit update requests to a name server. By definition, the name server sending update requests does not belong to the current server's domain and, therefore, cannot be a trusted host. There was, up to BIND 8.2, a security breach there.

7.8.2.2 DNS spoofing exploit

What use can hackers make of DNS spoofing? They can, for example, force an incorrect IP address as the translation of a given fully-qualified host name into the cache. This appears as just a routine update of the IP address associated with a given name, and nothing seems wrong with it. Since the complete qualified name is in the cache, there is not even a need to change anything in the hierarchy of DNS hosts involved.

Imagine they do that with one of your service providers, the HTML logon screen of which they have captured first. If they display it from one of their servers (the one you will access thinking you are connecting to your provider), you are going to type your account number and password right on the hackers machine without ever knowing it[9]. The hackers now just need to blackhole[10] you while they log on the service and usurp your identity. Since

[8] Which will certainly be the case if, at the same time, another attacking computer floods the real name server with irrelevant packets addressed to its MAC address (using the IP address would harm the fake DNS server as well).
[9] This will apply, of course, only if that service does not use time-based encryption, or does not update the encryption keys fast enough.

the operations used both your account number and your personal key, the service will probably not accept any responsibility for this usurpation.

To reduce that risk, a possible thing is to modify applications so they do an extra DNS lookup for each client. After translating the IP address of the incoming request to a host name, they use the DNS to translate from the supposed host name back to the IP address. If the two addresses do not match[11], the access is not granted and a warning is issued (faking an address in a direct access table is easy, but inserting the false address at the right place in a sequential reverse access table is more difficult).

To address the growing problem of IP spoofing, the RFC 2065 defines DNS Security Extensions (DNSSEC). These security extensions allow resolvers and name servers to cryptographically authenticate the source and guarantee the integrity of DNS queries and responses with digital signatures. They are part of BIND 8.2.1 and incorporated in AIX 4.3.3.

7.8.2.3 DNS spoofing compared to IP spoofing

A telephone comparison helps in understanding the difference between DNS spoofing and IP spoofing: with IP spoofing, a skilled technician just takes over the identity (IP number or phone number) of whatever has to be called. DNS spoofing is more subtle: the technician keeps his/her original number unchanged, but simply modifies a phone book somewhere in order to be called whenever a client using that phone book tries to call a given server.

DNS spoofing does need some IP spoofing at the very beginning to fake the identity of a DNS server. But once it is done, the hacker can revert to his/her original IP address (more precisely, the IP address of another site he/she is hacking), so nothing will seem abnormal on the network.

BIND 8.2 provides good built-in securities to prevent IP spoofing but, unfortunately, cannot enforce their use by network administrators.

7.8.3 DNS security tips

Consider the following DNS security tips:

- Recursive DNS commands allow hackers to get the complete list of hosts on your domain. Turn off recursion in DNS if you can (`recursion no`). If you cannot, consider these questions:
  - Which addresses are allowed to make recursive queries?

[10] Blackholing means intentionally giving no answer to a request. This technique is intended to consume time from the requester and is used both by hackers on their victims and by system administrators on potential hackers.

[11] For speed reasons, converting names to IP addresses and IP addresses to names do not use the same tables; this is why coherence is not always guaranteed.

- On which zones can they be given information?

Then get the BIND 8.2.1 upgrade from IBM. It allows you the selective allow-recursion substatement.

- Queries on authoritative zones coming from the outside must, of course, be answered because this is the responsibility of a name server. But queries outside the authoritative zones should never be accepted from any address not belonging to your domain. Enforcing this is not possible with bind4 (IPv4), but it is in bind8 (IPv6) with allow-query.
- The feature of DNS that allows it to update addresses in its cache is called glue fetching and is an open door to DNS spoofing. Turn glue fetching off (`fetch-glue no`). That will slow the DNS server, but it will help you avoid many potential troubles.
- Consider having two kinds of name servers: *advertising* name servers, open to the world and non-recursive, and *internal resolution* name servers that can only be queried by known resolvers.
  - Fetch-glue and recursive queries should only be allowed on the internal resolution servers.
  - In addition, recursive queries should only be allowed from a list of internal sources that are trusted.

7.9 X-Windows security

Let's first recall some X-Windows terminology. As seen from X-Windows, your Xstation, Network Station, or software on an AIX machine is called a server.

This seems strange at first, but there is an implacable logic behind it (remember that the X system comes from MIT). Your server is designed to honor requests from the outside world, just like your telephone. Moreover, unlike your telephone, it is able to honor many requests coming from many places at the same time as you can see in Figure 15 on page 166. This makes a deep security difference between an X server and a terminal as we will see.

Applications running on one or many host systems are the *clients* of your server. As such, they are the ones taking initiatives. Your server cannot do anything if there is not a client application (for example, aixterm) asking it to.

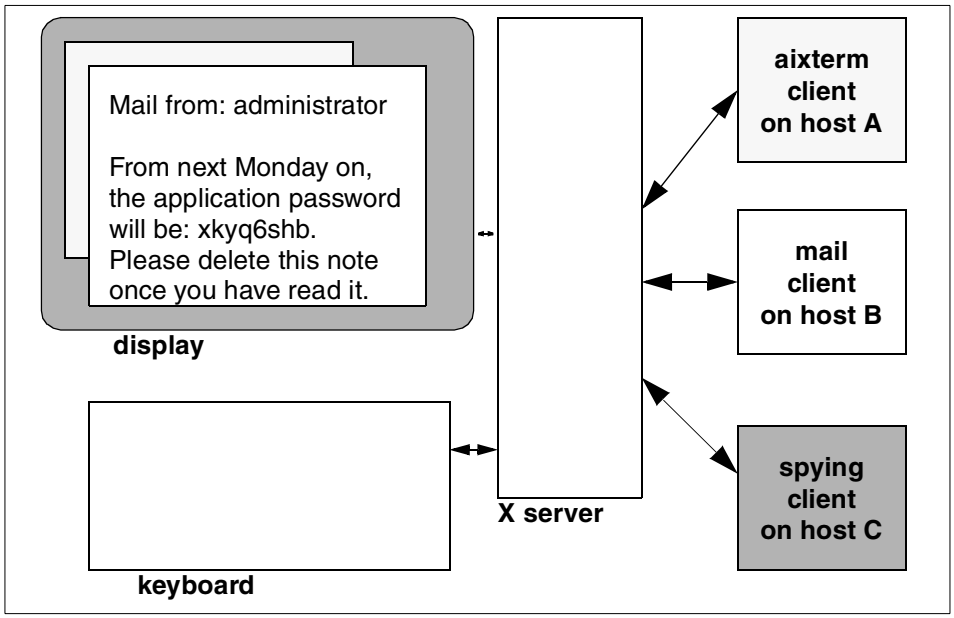

Figure 15. An X server used by the outside world

7.9.1 Remote spying

Your server is virtually connected to your local network (or even to the Internet) like a telephone is to the world phone network. Does that mean that any application on the network can use it? We should differentiate between two cases: your server is an AIX workstation or your sever is not an AIX workstation.

7.9.1.1 If your server is an AIX workstation

Everybody probably had at one time or another a message like this while trying to open a window on their workstation from a remote host:

```
Xlib: connection to "host_b:0.0" refused by server
Xlib: Client is not authorized to connect to Server
```

These messages mean that your server on an AIX workstation cannot be accessed without your explicit authorization. This means, among other things, that your manager cannot open a window on your screen every 10 minutes to tell you “Please come immediately to my office”. This can be a good thing if you need to concentrate on your work.

Removing the protection

But right now, you want to open that window on your station. In such a case, the first reaction of most of us is to type the command:

```
xhost +
```

So, every client application is allowed to connect to your graphic server. After all, should there be any harm in letting your display be accessible to anybody? Of course, anybody can display a window, but if you do not like it, you just kill it (but, what about a DOS attack opening so many windows your server crashes?). Well, at least they can only display things, not read them, right?

Wrong. Don't forget that your display is a *server*. Therefore, if you type `xhost +`, it now has to answer to any request made by any host. And the following requests made by an hostile host are a door to big information leaks:

- Dump the screen contents to a file.
- Get a list of the displayed windows.
- Monitor which client applications are using your display.
- Dump the contents of a specific window to a file.
- Capture keyboard keys.

All these things can be done by C programs using the X-Windows library. Worse, dumping your screen to a file does not even require any programming skill from your attacker. The `xwd` command allows anybody to do that from anywhere, and you will not even know it. As the IBM documentation says: "This file can then be read[12] by various other X utilities that perform functions such as redisplaying, printing, editing, formatting, archiving, and image processing" (from *AIX Version 4.3 Commands Reference*, SBOF-1877). And a cron program on the attacker's site can dump the contents of your screen every minute to store in a database for later exploitation.

For that reason, you should always specify the list of client applications and hosts authorized to access your station. In other words, `xhost +` should never be used in a form that has no argument. But a set of commands in sequence limits the risk a little, especially if you issue `xhost -host_a` and `xhost -host_b` as soon as you do not want to deal with these hosts anymore, for example:

```
xhost +host_a
xhost +host_b
```

[12] This file will be on the hacker's machine (note from the document authors).

7.9.1.2 If your server is not an AIX workstation

In such a case, you should check which are the default options of the particular software your system is running and, if they were installed before you were the system administrator, if these options were overridden or not by your predecessor

Important security issue

For easier installation, many X-server software packages for PC are shipped enabling connection by anybody as a default. You should realize that even though they are not running AIX, these workstations, if not reconfigured properly, do threaten the security of your AIX system, since anyone can capture your data the same way as previosly described.

We have a dilemma here. It is certainly wise not to use X-Windows on an AIX machine running an online banking application. But the comfort of X-Windows cannot be denied, and developers can hardly do any productive work nowadays without it. This certainly is an argument to have two physically distinct networks: a C2 network with X for the developers, and a regular network open to the outside world, without X, for production applications[13].

7.9.2 Local security: The xss command

The `xss` command uses the enhanced MIT screen saver extensions. As such, it will work on an AIX 4.3 workstation, may or may not work on PC servers depending on their version, and will not work on X servers using older versions of AIX (which must use the older xlock).

Prior to xss, users issued an `xlock` command when they left their workstation for some time. The `xlock` command is just an X version of the `lock` command used to lock an ASCII terminal (see 2.5.2, “Locking your terminal” on page 33). At a time when all PC users already had password-enabled screen savers automatically launched by time-outs, the `xlock` command began to appear outdated.

The `xss` command executes any command given as its argument when a screen saver time-out message is received, so you can use it as a mere screen saver, but also do some automatic security housekeeping whenever you are away from your workstation longer than you expected. The `xss` command is a mere local (physical) security for your workstation and does not

[13] What about a corporate Web server? Obviously, it has to be connected to the outside world, but can a Web server run without X-Windows? Yes, it can. Remember that HTTP servers are only *client* applications from your X server point of view, and, therefore, do not have to be X-enabled. And that from the HTTP point of view, the Netscape Navigator that uses your X *server* is just their *client*. Everything is relative.

address the problem of remote spying seen in the preceding section. To learn more about xss, refer to *AIX Version 4.3 Commands Reference*, SBOF-1877.

7.10 AIX Fast Connect for Windows security

NFS, the most common way to share files, uses IP. To share files between AIX hosts and PCs running Windows, the only solution until recently was to install PCNFS on the PCs so they could become NFS clients. In addition, the PCs could also be defined as a client of the AIX print server(s).

AIX Fast Connect for Windows takes a simpler approach—instead of installing a NFS client and customizing TCP/IP on scores of PCs, it installs the Microsoft networking protocol, called *Server Message Block*, (SMB)[14], on an AIX server.

For PC users, there is no strong difference. In each case, they have seamless access to AIX printing and disk resources with automatic backup facilities. Disk access with Fast Connect is, however, faster than with PCNFS, because SMB is a lighter protocol than IP.

AIX Fast Connect does provide UNIX-level security in the sense that the standard file protection securities (owner/group/other combined with read/write/execute) will be honored. However, it does not honor the access control lists (ACLs), which give better access granulation.

Also, the AIX ulimit and quota limitations are not checked, so there is a danger of going disk full. For this reason, it is wise to define these file systems on dedicated logical volumes.

Finally, there is no log information about the users connected via AIX Fast Connect. This is worth noting—take extra care in securing these machines.

To learn more about AIX Fast Connect for Windows, refer to Chapter 11, “AIX Fast Connect for Windows” in *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127.

[14] The most recent version of SMB has been renamed Common Internet File System, or CIFS. CIFS is approved by the Internet Engineering Task Force.

Chapter 8. Security management

The more computers you have, the more complex managing your security environment becomes. Some vendors may recommend purchasing third-party software to manage security across your company. As with any form of enterprise software, we have to recommend a careful evaluation before a roll-out. It would be simplistic to trust what has been written in marketing brochures. In this chapter, we go through some of the facilities provided by AIX that are used for security management.

8.1 Accounting and logs

System accounting gives you the ability to monitor CPU utilization, disk usage, and connection time. Monitoring these, you can determine if you have been penetrated by a sudden change in a user's status or utilization.

Some files are created even without turning accounting on, and these should be monitored regularly. A hacker may realize that these files exist and try to edit them to remove references to him or her from them, but auditing would catch the hacker (if you were running it).

Here is a description of the files that get updated without turning accounting on.

/var/adm/sulog
Owner is root, group is security, and permissions is 600. This is a log of all attempts to `su` from one user to another. It is a plain text file. Here is a sample:

```
SU 05/01 15:23 + pts/4 root-joe
SU 05/03 09:52 - pts/2 jane-joe
```

Note that this is "first event, first displayed". Here, you have root successfully becoming joe on May 01, at 15:23 on tty pts/4. Also, on May 03, at 09:52, jane failed to become joe. Perhaps you need to call jane (or her manager) and talk about using other's accounts. It depends on your security policy. We recommend reviewing this weekly, then clearing the file.

/var/adm/wtmp
Owner is adm, group is adm, and permissions is 664. This is a log of "all" the network connections. It is a data file with the format specified in utmp.h. The following is an example:

```
# last joe
joe ftp   host_a May 11 13:42 - 13:44  (00:02)
joe pts/1 host_b  May 10 13:32 - 13:38 (00:05)
```

```
wtmp begins     May 02 11:37
```

Note that this is “most recent displayed first.” Here, we see that joe connected via ftp from host_a on May 11 for two minutes, and he was connected to a tty from host_b on May 10 for five minutes. We recommend reviewing this file once a week and then clearing it.

/etc/utmp
Owner is root, group is system, and permissions is 644. This has “all” the current connections to ttys in it. It is a data file, with the format specified in utmp.h. It can be easily read with the `who` command. For example:

```
# who
joe      pts/0       May 11 16:52    (leo)
# w
  10:53AM   up  18:13,  1 user,  load average: 0.00, 0.00, 0.00
User     tty          login@       idle      JCPU      PCPU what
joe      pts/0       04:52PM          0        41         0 w
```

You can see that joe logged in from leo on May 11 at 4:52 PM. No one else is logged in. This output is sorted by tty.

/etc/security/failedlogin
Owner is root, group is system, and permissions is 644. This is a data file with the format specified in utmp.h. For example:

```
# who /etc/security/failedlogin | tail
UNKNOWN_    pts/1       May 12 11:28    (loopback)
jane        pts/1       May 12 11:28    (loopback)
jane        pts/1       May 12 11:30    (loopback)
jane        pts/1       May 12 11:30    (loopback)
UNKNOWN_    pts/1       May 12 11:52    (loopback)
```

User `UNKNOWN_` means that someone tried to connect with an invalid user name. If you removed the lpd user, and someone tried to connect as lpd, it would look like this.

/etc/security/lastlog
Owner is root, group is security, and permissions is 640. This is a plain text file that has a stanza for each user that has logged in (or attempted to log in). For example:

```
# grep -p jane /etc/security/lastlog
jane:
        time_last_login = 958145320
        tty_last_login = /dev/pts/1
        host_last_login = loopback
```

```
unsuccessful_login_count = 3
time_last_unsuccessful_login = 958145405
tty_last_unsuccessful_login = /dev/pts/1
host_last_unsuccessful_login = loopback
```

The references to time are in UNIX time (seconds since Jan 1, 1970). The stanzas are well documented in the file itself. In this case, jane tried to connect to loopback. We recommend monitoring this file for high unsuccessful_login_counts. This might show someone trying to brute force break someone's password. Disable the account if necessary until the real user calls to have it fixed.

If you want any more accounting, you have to turn on accounting and actually run reports. This is thoroughly discussed in *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126.

We recommend that you familiarize yourself with the accounting procedures, records, and reports. This will give you the ability quickly to turn accounting on and get a record of activities on your system.

8.2 Security backup

Backups are an important part of security. Without backups, it is very difficult to track down what has changed and recover from an incident. Therefore, it is important to have a backup policy before a system is in operational use. Most enterprise has a company-wide backup policy to follow. There is also a current trend toward the use of Storage Area Network (SAN) for data storage management. This document does not cover SANs, instead we focus on some general backup issues before zooming in on specific AIX considerations. We recommend Chapter 9, "Backup and Restore" in the *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126 for more details about backup configuration and commands.

The following are some backup objectives:

- Backups allow the recovery of files that have been modified, added, or deleted.
- Backups allow comparisons between files at different times to see what has been changed.
- Backups of system logs or audit trails can be used for investigating incidents and recreating what has been done to the system.
- Backups can be used as evidence in court.

There are some issues to keep in mind when planning for a backup and we cover each briefly in the following sections.

8.2.1 Backup media

The security of the backup media is extremely important. If the backup contents are not properly protected, it becomes the weak link in your security defense and can be prone to attack. As an example, if a backup is done to a tape device, ensure that only authorized accounts can read and write this tape device. Besides logical security, physical security must be controlled as well to prevent unauthorized people from stealing or duplicating the tapes.

Some backup media are more secure than others. These are the ones that can only be written once. Examples range from a printer that continuously prints out log messages piped to it or logging to a mounted CD-R device. This is obviously only suitable for specific situations or when an incident is suspected. Most people make use of tapes or rewritable media in the normal course of a backup.

Backup media is best stored at a separate location from the systems. This prevents the simultaneous destruction of both backups and systems when the physical location is affected by disasters, such as floods, fire, or terrorist attacks.

8.2.2 What do you backup?

There is a trade-off between storage costs and the amount you want to backup. The management of large amounts of data can also be unwieldy.

Audit, accounting, and system logs
Remember that you have to install and turn on accounting and audit if required. Syslog is installed by default but does not log to any files. Edit the /etc/syslog.conf to log for more details. By default, no syslog logging is output to file. We recommend that at least *.crit are logged to file for backup.

Table 16 summarizes the logs that you may consider backing up.

Table 16. Audit, accounting, and system logs

Files or directories	Description
/etc/security/lastlog	Information for user login. ASCII format.
/etc/security/failedlogin	Contains entry for failed logins. Read by who.
/var/adm/cron/log	Cron output logs. ASCII format.
/var/adm/sulog	Logs su attempts. ASCII format.

Files or directories	Description
/audit/trail	AIX audit (BIN mode) trail directory. The events and objects in the audit configuration file defines what is audited. Read by `auditpr`.
/var/adm/wtmp	AIX accounting file. Contains connect-time accounting data, including login, logoff, and shutdown records. Read by last, runacct, and fwtmp.
/etc/utmp	Contains active users and subsystem. Read by who.
/var/spool/mqueue/log	sendmail log. ASCII format.

Configuration files

What files are important depends on what is being run on the system and what the system is being used for. Besides the log files in the preceding section, there are also important configuration information that you may consider backing up. Table 17 shows essential configuration files that you may want to back up. This list is by no means complete. There are additional configuration files depending on the applications installed.

Files may be backed up individually or as part of a file system. There are some users who prefer backing up individual files. One reason is the ability to quickly extract the information anywhere. If the backup is done by tar, the files can be extracted in any UNIX system. PC versions of tar are also available. You may consider using doswrite, which is available on AIX to write to floppy. You may also consider commercial backup software that is usually useful in a heterogeneous environment.

Table 17. Essential configuration files

Files	Description
/image.data	Information of image installed.
/etc/.rootkey	Key for Secure NFS operation.
/etc/aliases	Alias definition for sendmail.
/etc/dumpdates	File system backup information.
/etc/environment	System-wide environmental variable definitions.
/etc/hosts	Name-to-IP-address mapping information.
/etc/hosts.equiv	Remote systems that can execute commands on the local system.
/etc/gated.conf	Gated configuration file.

Files	Description
/etc/gateways	Route information for routed.
/etc/group	Basic attributes of group.
/etc/filesystems	File system characterizations.
/etc/ftpusers	Local user names that cannot be used by remote FTP clients.
/etc/keystore	Key for Secure NFS operation.
/etc/inetd.conf	inetd configuration file.
/etc/motd	Message-of-the-day file.
/etc/netsvc.conf	Ordering of name resolution services.
/etc/networks	Networks of NFS Internet.
/etc/ntp.conf	xntpd configuration file.
/etc/ntp.keys	Authentication keys for xntpd.
/etc/passwd	Basic user attributes.
/etc/profile	System-wide environment customization.
/etc/publickey	Public or secret keys for maps.
/etc/qconfig	Printer queuing system configuration.
/etc/resolv.conf	Name-server information for local resolver routines.
/etc/rpc	Database for RPC program numbers using NFS.
/etc/sendmail.cf	sendmail configuration file.
/etc/syslog.conf	syslogd configuration file.
/etc/snmpd.conf	snmpd configuration file.
/etc/telnet.conf	telnetd configuration file.
/etc/vfs	Defines VFS installed.
/etc/security/login.cfg	Configuration information for login and user authentication.
/etc/security/environ	Environment attributes of users.
/etc/security/group	Extended attributes of groups.
/etc/security/limits	Process resource limits of groups.

Files	Description
/etc/security/passwd	Password information.
/etc/security/pwdhist.dir /etc/security/pwdhist.pag	Password history information.
/etc/security/portlog	Per-port unsuccessful login attempt information and port locks.
/etc/security/roles	List of valid roles.
/etc/security/smitacl.group	Group ACL definitions for SMIT.
/etc/security/smitacl.user	User ACL definitions for SMIT.
/etc/security/sysck.cfg	File definitions for trusted computing base.
/etc/security/user.roles	List of roles for each user.
/etc/security/user	Extended user attributes.
/etc/security/audit/config	Audit system configuration information.
/etc/security/audit/objects	Audit events for audited objects.
/etc/security/audit/events	Audit events of the system.
/etc/security/audit/bincmds	Auditbin backend commands.
/etc/security/audit/streamcmds	Auditstream commands.
/usr/lib/security/mkuser.default	Default attributes for new users.
/var/adm/cron/cron.allow	Users allowed to use crontab.
/var/adm/cron/cron.deny	Users not allowed to use crontab.
/var/adm/cron/at.allow	Users allowed to use at.
/var/adm/cron/at.deny	Users not allowed to use at.

8.2.3 How do you backup?

Several commands shown in Table 18 on page 178 create backups and archives. Because of this, data that has been backed up needs to be labeled as to what command was used when doing the backup and how the backup

was made (by name or by file system). The `backup` command is the most frequently used, but other commands serve specific purposes.

Table 18. Backup commands

Command	Description
backup	Backs up files by name or by file system.
mksysb	Creates an installable image of the rootvg volume group.
cpio	Copies files into and out of archive storage. Can usually read data archived on another platform provided it is in cpio format.
dd	Converts and copies a file. Commonly used to convert and copy data to and from non-AIX systems, for example, mainframes. dd does not group multiple files into one archive; it is used to manipulate and move data.
tar	Manipulates tar archives. Useful for backing up directories and files. Readable from non-AIX systems.
rdump	A network command that backs up files by file system onto a remote machine's device. *Not recommended*. The use of rdump establishes a trust relationship with poor authentication.
pax	POSIX-conformance archive utility that can read and write tar and cpio archives.

One of the reasons for backups is to examine files or file systems on a separate system. If the idea is to clone another identical system from the backup, `mksysb` is the obvious choice. `backup` is useful for user files and file systems. However, if you would like to have the flexibility of examining log files on a different OS, `tar` and `dd` can be used. Note that if you are planing to back up the audit trail, they are in binary format and require another AIX.

8.2.4 When do you backup?

Another important question is when to back up. Your backup policy should categorize the different machines in your organization and their backup requirements. From a security viewpoint, backups are required to either restore files or scan for log entries; therefore, backups should be of a sufficient period to cover your computer audit periods.

We recommend backing up at least once a week and keeping the backups for eight weeks. However, this may be unreasonable in some circumstances. Back up your system before and after you make system changes.

8.3 Intrusion detection and recovery

In this section, we cover basic intrusion detection concepts and how AIX relates to intrusion detection and recovery.

Before going into AIX-specific details, there are a few points to note when we discuss intrusion detection.

Intrusion detection has two purposes. The first purpose is to notify about a potential intrusion or an intrusion that has already taken place. Intrusion detection is analogous to a burglar alarm in the real world. The second purpose is the ability to create a profile of your security threat. Over time, such a profile allows more informed purchasing and budgeting decisions to be made on security products. Intrusion detection does *not* prevent intrusions. This is an important point to recognize. There are actions that can be automated after a certain attack signature is detected. But in general, the protection of systems comes from good host security practices and firewalls. A convergence of some of this functionality might occur but whether it makes sense to put all the eggs in one basket is debatable.

What action do you take when the intrusion is detected? An effective intrusion response policy must be in place for an effective intrusion response. As an example, if the machine is providing a service and generating revenue from it, the decision to unplug the machine should not lie in the hands of the systems administrator making an ad-hoc decision. This is a business decision that needs to be decided in the incident response policy beforehand.

Another critical point of intrusion response is “what is considered an intrusion”. Different organizations have different definitions. If your network gets scanned everyday from the Internet, is that an intrusion and what action do you take?

Placement of an Intrusion Detection System (IDS) in a network is critical. If it is in a place exposed to the Internet, a large number of alerts are gathered because of the constant scanning that goes on. Collection of such data over time is useful for getting a profile of what is probing your network or the effectiveness of an edge security device. Unfortunately, the numerous alerts create huge logs and have a tendency to cover up more serious attacks. Alerts need to be classified in terms of priority and associated actions.

An intrusion response policy allows decision makers to come to a consensus on what constitutes an attack and the response required. Intrusion response requires more than a technical solution. Departments, such as human

resources or legal, and company spokespersons may also be involved to deliver a coherent and effective response.

The role that AIX can play has two folds. First, AIX can be the platform running an IDS. In this role, the AIX is like a watchdog on the network, sniffing out potential intrusions and alerting you of them. The emphasis is on the network rather than AIX itself. AIX does not come with a network IDS by default and has to be installed separately. Alternatively, by the use of AIX software, such as iptrace or tcpdump, and some custom shell scripts, it is possible to create a simple network IDS.

Second, AIX can detect intrusions on itself. The following methods can be used in intrusion detection:

- Trusted computing base (TCB)
- UNIX logs
- AIX audit
- AIX accounting
- AIX system commands
- AIX network commands
- Network sniffing

Look for suspicious activities by users or daemons. It is important to maintain the integrity of the logs. Intruders who have obtained root access are able to modify or delete logs to hide their tracks. Throughout this chapter, we also use AIX commands. It is important that the integrity of these files be assured. Signed copies of these files stored offline in a safe, physical location is always a good practice. At the least, we recommend going into SAK to ensure the usage of trusted programs (refer to 6.2.6, “Secure Attention Key (SAK)” on page 117).

AIX installed with TCB provides a good intrusion detection mechanism by checking for changes against a baseline. The most important point when using tcbck is to ensure the integrity of the tcbck and sysck.cfg files. Users who are security conscious may want to store the signed copies of this two files offline to ensure their integrity. Of course, trusted programs that are added to the sysck.cfg must be pristine. Refer to Chapter 6, “Trusted computing base (TCB)” on page 109 for more details.

Besides the TCB features, AIX has the usual system and application logs. syslog.conf needs to be modified to create more logging. Table 16 on page 174 shows the important log files you can check and track usage, last logins

and logouts, and su logs. Logging is covered in 8.1, “Accounting and logs” on page 171. The cron log files may also provide useful clues if the intruder has installed programs to run by cron.

The following shows a screen output for the `last` command. Notice that it shows useful information, such as user name, source IP, date, and activity:

```
reboot    ~                               Apr 28 11:43
shutdown  pts/2                           Apr 28 11:39
root      pts/3        9.12.0.124         Apr 28 11:10 - System is halted b)
root      ftp          9.12.0.124         Apr 28 11:09 - 11:09  (00:00)
root      pts/2        9.12.0.128         Apr 28 10:42 - System is halted b)
root      pts/13       9.12.0.129         Apr 27 16:51 - 19:43  (02:52)
root      ftp          9.12.0.124         Apr 27 16:50 - 16:51  (00:00)
root      pts/8        9.12.0.129         Apr 27 16:48 - 20:11  (03:23)
katie     pts/13       9.12.0.128         Apr 27 16:28 - 16:28  (00:00)
root      pts/4        9.12.0.128         Apr 27 15:39 - 20:11  (04:32)
root      pts/4        9.12.0.128         Apr 27 15:37 - 15:38  (00:00)


wtmp begins     Apr 26 15:59
```

AIX audit has the capability to audit a large number of events and objects. This creates a large amount of detailed information that may be useful for deeper investigation. The trade-off is that such details occupy a large amount of space and are harder to interpret.

AIX accounting, while mainly used for billing purposes, may also be used for determining usage patterns. Accounting is covered in 8.1, “Accounting and logs” on page 171.

The AIX *ck commands `grpck`, `userck`, and `pwdck` are useful. They can be used to verify group membership, user ID definitions, and authentication stanzas, respectively. See 3.3, “Verifying the user environment” on page 54 for more information. The `find` and `ncheck` commands can be used to locate hidden/setuid/setgid files that are frequently used by intruders during an attack.

Another suite of commands are the AIX network commands. They are useful in checking network statistics and usage. We recommend looking at `netstat` outputs for both routing and service information. You may also determine the type of DDOS attack you are under using the `netstat` command.

The last part for intrusion detection is network sniffing. This can be performed using `iptrace` and `tcpdump`[1]. This brings the interface to promiscuous mode and sniffs the network traffic. There has been some discussion on whether it makes sense to have the capability to go into promiscuous mode, potentially helping an intruder that has broken into the system to propagate to other

[1] You can either use iptrace or tcpdump at any one time. tcpdump outputs can be examined on other UNIX and several network analysis software or hardware sniffers.

machines on the network. However, /dev/bpf? is only readable by root, and assuming that the intruder has already obtained root access, it is a matter of time before the device can be brought to promiscuous mode. Sniffing is especially useful across gateways to track packet movements. For related information, see CERT advisory CA-94:01 available at:

```
http://www.cert.org/advisories/CA-94.01.ongoing.network.monitoring.attacks.html
```

The final point to take care of when looking at different logs and sniffer outputs is the time. Most investigations take place across several machines and it is important to synchronize the time between them. This can be done manually, or preferably, through the use of xntp[2] (using authentication features). The trade-off is that an additional service is required, potentially introducing a new point of attack.

Note that all action taken during the course of an investigation should be in accordance with your organization's policies and procedures. We do *not* recommend that attacks be made against an offending IP. IPs can be easily spoofed or the IP is just another victim machine. This is still a hotly debated issue and the legal consequences are not known. A good practice is to establish good relationship with your Internet Service Providers (ISPs). They are able to filter offensive sources and track back the hops. Using the who is database and domain name lookups are great for intelligence gathering.

[2] Do not use the original ntp protocol, which does not provide authentication.

Chapter 9. Security products and software

This chapter provides you with information on security products that you can purchase from IBM or other retailers.

9.1 IBM security products

Having established the RS/6000 family of platforms as a solid foundation upon which to build a business, it is time to discuss the upper layers of what that foundation supports. Seldom is a platform purchased for its operating environment alone, and as secure as AIX is, middleware and solutions available for AIX are also key elements to a functionally useful, operationally secure environment.

Technologies that further enable the AIX environment as a platform on which secure applications can be built include:

- IBM 4758 PCI Cryptographic Coprocessor
- Tivoli SecureWay Public Key Infrastructure
- IBM SecureWay Directory
- IBM DCE for AIX Version 2.2

9.1.1 IBM 4758 PCI Cryptographic Coprocessor

The IBM 4758 PCI Cryptographic Coprocessor is a state-of-the-art, tamper-responding, programmable cryptographic PCI card that offloads the system unit's processor of complex and numerically-intensive cryptographic operations. The card handles Secure Electronic Transactions (SET) and RSA private key transactions and uses the IBM Common Cryptographic Architecture or other APIs to access services.

The following are its security features:

- Provides mechanisms for detecting specified physical penetration, voltage, temperature, and radiation attacks.
- Includes time-of-day clock for time/date stamping, as well as electronically-protected external memory bus and RS-232 electronics.
- Offers a noise-based true random number generator so that values are unpredictable.

9.1.2 Tivoli SecureWay Public Key Infrastructure

Tivoli SecureWay Public Key Infrastructure is specifically designed to integrate with backend systems, thereby helping, protecting, and leveraging the systems already in place. Tivoli SecureWay Public Key Infrastructure includes the following functions:

- A trusted Certificate Authority (CA) manages the complete life cycle of digital certification. To vouch for the authenticity of a certificate, the CA digitally signs each certificate. It also signs Certificate Revocation Lists (CRLs) to vouch for the fact that a certificate is no longer valid. To further protect its signing key, you can use cryptographic hardware, such as the IBM SecureWay 4758 PCI Cryptographic Coprocessor.
- A Registration Authority (RA) handles the administrative tasks behind user registration. The RA should be used to help ensure that only certificates that support a company's business activities are issued and that they are issued only to authorized users. The administrative tasks can be handled through automated processes or human decision-making. Policy exits enable application developers to customize the registration processes according to a company's defined policies and procedures.
- A Web-based enrollment interface makes it easy to obtain certificates for browsers, servers, and other purposes, such as virtual private network (VPN) devices, smart cards, and secure e-mail.
- An Audit subsystem maintains audit records, encrypted for security, for important transactions. It computes a message authentication code (MAC) for each audit record. If audit data is altered or deleted after it has been written, the MAC is used to detect intrusions. Audit records may also be signed when archived.
- The IBM SecureWay Directory is used to store certificates and revoked certificate lists in an LDAP-compliant format. By using IBM DB2 as its storage facility, the SecureWay Directory is highly scalable and has been tested with over 30,000,000 entries.
- The server components maintain separate databases for configuration data, registration data, certificate data, audit data, and directory data. Using DB2 offers extensive security features and storage capacity. For example, it enables Tivoli SecureWay Public Key Infrastructure to store registration data in an encrypted format and to perform integrity checks on stored audit records.

9.1.3 IBM SecureWay Directory

IBM SecureWay Directory provides a common directory for customers to address the proliferation of application-specific directories, a major driver of

high costs. IBM SecureWay Directory is a Lightweight Directory Access Protocol (LDAP) cross-platform, highly scalable, robust directory server for security and e-business solutions.

The key new feature provided with the IBM SecureWay Directory 3.1.1.5 is the support for DB2 Version 6.1. The IBM SecureWay Directory utilizes the IBM industry-leading DB2 database technology as the backend data store. Utilizing DB2 as the backend data store provides:

- Industry-leading search performance
- Scale for large directory size
- Proven 24x7 reliability, management, and database technology

IBM SecureWay Directory 3.1.1.5 now supports using DB2 Version 5.2 or 6.1 as the backend data store. You may continue to use DB2 Version 5.2 or upgrade to DB2 Version 6.1.

9.1.4 IBM DCE for AIX Version 2.2

IBM DCE for AIX, Version 2.2 is based on the Open Software Foundations DCE Version 1.2.2 release. The following are significant items that have been added since the IBM DCE for AIX, Version 2.1 release. Significant items added by OSF in the DCE 1.2.2 release include:

- Kerberos V5 support.
- The DCE security service includes an implementation of the MIT Kerberos Version 5 (V5) authentication and key distribution service.
- Public key support.
- DCE 1.2.2 allows public key technology to be used to support login. With this technology, the security server does not need to store the long term key (or password) for a principal so that it will remain undisclosed should any compromise of the security server occur.
- User-to-user authentication.
- Global groups.
- DCE 1.2.2 allows principals from a foreign cell to be added to groups in the local cell. Tasks, such as enterprise-wide security administration, cell reconfiguration, and other management tasks, are also made much easier.

9.2 Third-party software

There is a large amount of security software on the market. Your choice of software will depend on your budget, human resources, and company policies.

Note

There is a difference between running security software on AIX and securing the AIX machine itself. We do *not* recommend security software to be installed on the AIX server you are trying to protect unless it is necessary (for example, file integrity checkers).

The software listed is just provided as a sampler of what is available. Both commercial and non-commercial software are listed. Commercial software is denoted by "$" and non-commercial software is denoted by "-" in their respective rows. "$-" means it is available for both commercial and non-commercial products. Remember to verify the digital signatures or checksums if available. No guarantees are made on their compatibility or impact with AIX.

9.2.1 Security scanners

Scanners are useful in detecting systems and network vulnerabilities. They allow multiple hosts on the network to be scanned automatically. Install and run scanners from a separate machine[1] and connect the machine from the network only when running scans. This is to prevent a hijack of your scanning machine. We recommend running scans on your systems before connecting to any public or external networks, such as the Internet. In addition, regular scanning can be performed as part of an security audit process. See Table 19.

Table 19. Security scanners

Product name	$/-	Web site
nmap	-	http://www.insecure.org/nmap/
nessus	-	http://www.nessus.org/
satan	-	http://www.fish.com/~zen/satan/satan.html
NAI Cybercop	$	http://www.pgp.com/asp_set/products/tns/ccscanner_intro.ap

[1] Scanners do not necessarily need to be run on AIX. In fact, many commercial scanners may only support a limited number of OSs.

Product name	$/-	Web site
ISS Security Scanner	$	http://www.iss.net/securing_e-business/security_products/security_assessment/system_scanner/index.php
strobe	-	ftp://suburbia.net:/pub/strobe.tgz
netcat	-	http://www.l0pht.com/users/10pht/nc11nt.zip http://www.l0pht.com/users/10pht/nc110.tgz

9.2.2 Intrusion detection

The term Intrusion Detection System (IDS) is used loosely in this section to represent any software that has the capability to aid in detecting intrusions. Most commercial IDSs will have a more stringent definition of what IDSs are and their capabilities. More details are available in 8.3, “Intrusion detection and recovery” on page 179.

9.2.2.1 Host-based IDS

Log management software is the simplest and cheapest form of detecting intruders. It is possible to run custom shell scripts on log files to detect for anomalies, but log management software abstracts this to a higher level and makes it easier to check for anomalies. See Table 20.

Table 20. Intrusion detection (Host-based log management)

Product name	$/-	Web site
logsurfer	-	http://www.cert.dfn.de/eng/logsurf/
swatch	-	ftp://ftp.stanford.edu/general/security-tools/swatch

Another class of host-based IDSs are the file system integrity checkers. Note that AIX already provides tcbck, which preforms the same function. We recommend the use of TCB on AIX.

Table 21. Intrusion detection (Host-based file system check)

Product name	$/-	Web site
tripwire	-	ftp://ftp.cert.dfn.de/pub/tools/admin/Tripwire/
	$	http://www.tripwire.com/
l5	-	ftp://ftp.cert.dfn.de/pub/tools/admin/L5/

9.2.2.2 Network-based IDS

Intrusion Detection Systems (IDS) are useful in alerting users to intrusions occurring at the network or host level. They also provide a good

measurement of the attack you are facing over a period of time. Currently, most network IDSs make use of attack signatures and sniff the network for occurrences.

Table 22. Intrusion detection (Network-based)

Product name	$/-	Web site
snort	-	http://www.clark.net/~roesch/security.html
nfr	$	http://www.nfr.com
Network ICE	$	http://www.networkice.com/
ISS Realsecure	$	http://www.iss.net/

9.2.3 Encryption

Encryption allows the traffic session or file to be read by authorized users. More recently, the idea of virtual private networking (VPN) has gained acceptance as a means of remotely passing confidential traffic across the Internet as a means of cutting international communications costs. AIX has the ability to tunnel traffic using IP Security (refer to 7.3, “IP Security” on page 136). This is a powerful feature and we recommend its use.

Table 23. Encryption

Product name	$/-	Web site
pgp	$-	http://www.pgp.com/
ssh	$-	http://www.ssh.com/

9.2.4 Host security

Host security is an essential part of AIX. Proper configuration of AIX is the first and most important step in ensuring that AIX is secure enough to be deployed in a real-world environment.

Table 24. Host security

Product name	$/-	Web site
cops		ftp://ftp.cert.dfn.de/pub/tools/admin/Cops/
sps	$-	http://www.starsecure.com/

Chapter 10. Securing the AIX platform

This chapter takes a practical approach to what needs to be done to secure an AIX platform. We look at various ways that AIX may be compromised followed by an example of how to secure your AIX platform.

Parts of this document assume a certain level of AIX knowledge. The AIX documentation library has many resources that will be useful in the course of this chapter. See:

```
http://www.rs6000.ibm.com/resource/aix_resource/Pubs/
```

First, you need to evaluate your system and understand the risks you are taking. One way of doing this is the Threat-Vulnerability Analysis. You need to identify the threats and the kind of vulnerabilities that your system is exposed to. We do not complete a formal analysis here, but simply walk through examples of threats. At this point, it is useful to refer to Chapter 1, “Introduction” on page 1 again for the various aspects that go into a good security approach.

There are myriads of ways to getting unauthorized root access to a system, for example:

- Use a Trojan Horse on a careless administrator to create a back door.
- Use a known exploit on an unfixed system.
- Use a little known exploit on a “supposedly fixed” system.
- Use a new exploit on a “supposedly fixed” system.

There are exploits for local access, such as permission problems. There are also exploits for network access, such as service configuration errors. Since UNIX security is based on trust, it actually is easy enough to find exploits, and easy enough to disable them.

A “user” can convince (through guile perhaps) a system administrator to run a program that the user has written or modified to capture root’s password, create a SUID shell, install a backdoor, and so on. Once a user has root access, the user can install a “root kit” that will attempt to remove him or her from the process table, connection list, auditing files, accounting system, and so forth. The hacker can then even establish other accounts, or backdoors to your system. With this established, a hacker can do what ever they want to your system, and return later.

A hacker can convince certain network service programs to run files, too. The Common Desktop Environment (CDE) is notorious for having holes in it, so is

sendmail. These programs might be convinced to run either an interactive program, or perhaps, a predefined program by a hacker. They use this to install their backdoor, or create a root account.

A hacker can use IP spoofing, by telling everyone he or she is someone else, someone that you trust, and then walk right in your front door.

You might think that this is no big deal. Your severs do not have confidential material, nor are they trusted by any other network devices. However, someone could use your machines to attack another system. This might have serious consequences for your legal department. How would your clients respond to you losing a server to a hacker? They do not care about the servers insignificance.

There are other types of attacks that can happen to you as well. Attacks aimed at your network to disable your communications with other computers. These are called Denial of Service (DOS). Recently, there has been a new form of attack—hacking several computers and using them to mount a DOS on a different system. This has been dubbed Distributed Denial of Service (DDOS). The lawyers are still working out the legal issues of this. However, you do not want to have to fight a liability case in a courtroom.

There are several ways to prevent these various attacks. It all boils down to the four security principles mentioned in 1.1.2, “Goals of security” on page 4. Understanding and applying these principles is paramount in securing your systems. Here are a few examples of how to apply them:

- When helping a user, and you need to become root, make sure the user is not sniffing the network for traffic, make sure you are running the program you think you are running by specifying the full path, and make sure it is trusted. See Chapter 6, “Trusted computing base (TCB)” on page 109.
- Do not send your password in clear text over a network. Use IPSec, tunneling software (like secure shell), or other authentication methods, such as DCE. See Chapter 7, “Networks” on page 123.
- Install and configure a firewall.
- Remember that hackers are never satisfied with yesterdays exploits. They are always trying to find new ways to break into systems, or to bring them down. Hacking is constantly evolving and growing. Stay informed by doing the following:
  - Read the news at `http://rootshell.com`
  - Read the news at `http://www.cert.org`

- Sign up on the CERT mailing list. Send mail to `cert-advisory-request@cert.org` with the subject “SUBSCRIBE *your-e-mail-address*”

10.1 Overview

There are six steps in securing a platform and ensuring its validity during operational use:

1. Install and secure an AIX operating system (including fixes).
2. Install and secure applications (including fixes).
3. Install filters and/or IPSec.
4. Pre-deployment testing.
5. Operational deployment.
6. Regular monitoring.

We do not cover all of the above steps. Some steps, such as installation of IP filters, have already been separately covered. Instead, this chapter focuses on the installation and configuration of AIX, effectively hardening it. The first and most important step is to have a secure AIX platform to work on.

The requirement to harden a platform has already been recognized as an industry requirement. In the ITL Bulletin December 1999: OPERATING SYSTEM SECURITY: ADDING TO THE ARSENAL OF SECURITY TECHNIQUES by the National Institute of Standards and Technology (NIST), OS security technologies are recommended as part of any security plan. See:

`http://www.nist.gov/itl/lab/bulletns/dec99.htm`

In the same vein, The Ten Worst Security Mistakes Information Technology People Make by SANS Institute lists connecting systems to the Internet before hardening them as number one mistake. See:

`http://www.sans.org/mistakes.htm`

We walk through a typical process to create a secure platform. Note that this chapter is securing a sample AIX platform. Do not follow these steps without knowing their impact on the operational usage. It is a good idea to perform these lockdown procedures first on a test machine before actual roll out to live use.

It is good to keep in mind the security principles throughout the whole securing process. Whenever you remove a fileset or limit a functionality, a

decision needs to be made on the security versus convenience. An example is the use of Common Desktop Environment (CDE). While CDE is a very nice environment to work on, there are potential dangers from its use. In the end, a decision needs to be made on whether CDE is necessary or just nice to have. A simple way to look at it is the purpose of the system. Ask yourself what is the impact when you require a certain service. Is it necessary and is it worth the risk? We will be making some assumptions on what are needed in our examples, but the basic concepts and steps are still the same.

If there are security policies or standards in your company, follow them. If you feel that the measures are no longer adequate in the Internet age, let the policy or procedure writers know about it.

10.2 Installation

The installation process is the first and most important step in securing a system. We recommend a fresh installation (with TCB) wherever possible. If preservation mode is used, there are chances that insecurities from the previous system will be carried over.

Set password for root as soon as you can. Usually, the Installation Assistant launches when you first install a machine. You can set the password there.

If the default installation is chosen (only the BOS rte), there are still several filesets you may want to consider removing. Check the dependencies before removal. The more filesets or services removed, the more secure your system is.

perl.rte is an example of a fileset that provides many useful features. However, the presence of such a powerful tool as perl may not be a good idea if security is important.

Finally, you should remember to perform these tasks on a stand-alone machine rather than one connected to the network.

10.2.1 Removal of services

As the previous section emphasizes, do *not* install more filesets than necessary. Several daemons and services are started if you install the server filesets, such as:

- bos.net.tcp.server
- bos.net.nfs.server
- bos.net.nis.server

We do not recommend the installation of these filesets unless necessary. bos.net.tcp.server has a `securetcpip` command that disables extremely risky servers to be started from inetd. This disables tftp, utftp, tftpd, rcp, rlogind, rlogind, rsh, and rshd. This command is controlled by /etc/securetcpip.

On an IBM SP, you will need tftp and tftpd, so do *not* change its permissions. You can, however, leave the service stopped until it is time to install a machine, then explicitly start it (and stop it when you are done).

It is better to deliberately disable all your network services, then explicitly configure and enable specific services that are required by the applications and users.

One simple reason for this is that it will protect you from new exploits in services that you are not running. If a window is bricked in, a burglar cannot use it to sneak into your house. However, if you left a service in its default configuration, it might be a matter of time before it is broken into. The lock may claim to be safe from lockpicks when it left the factory but that doesn't mean that it will always be. And do we truly know what is the level of lockpicking skills in the world?

The Cuckoo's Egg, by Clifford Stoll (refer to C.3, "Other resources" on page 215) is a good story about what happens when you leave defaults unchanged. While today's defaults may be better than the defaults we had in 1986 (the time frame of Stoll's book), if someone finds an exploit for a default configuration, and you have it turned off, they cannot exploit your system.

The following shows the output of `netstat -a -f inet`. Each service listening (LISTEN) can be a potential security vulnerability:

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  *.websm               *.*                    LISTEN
tcp4       0      0  *.dtspc               *.*                    LISTEN
tcp4       0      0  *.32792               *.*                    LISTEN
tcp4       0      0  *.instsrv             *.*                    LISTEN
tcp4       0      0  *.time                *.*                    LISTEN
tcp4       0      0  *.daytime             *.*                    LISTEN
tcp4       0      0  *.chargen             *.*                    LISTEN
tcp4       0      0  *.discard             *.*                    LISTEN
tcp4       0      0  *.echo                *.*                    LISTEN
tcp4       0      0  *.32791               *.*                    LISTEN
tcp4       0      0  *.netstat             *.*                    LISTEN
tcp4       0      0  *.systat              *.*                    LISTEN
tcp4       0      0  *.finger              *.*                    LISTEN
tcp4       0      0  *.uucp                *.*                    LISTEN
tcp        0      0  *.exec                *.*                    LISTEN
tcp4       0      0  *.klogin              *.*                    LISTEN
tcp        0      0  *.login               *.*                    LISTEN
tcp4       0      0  *.kshell              *.*                    LISTEN
tcp        0      0  *.shell               *.*                    LISTEN
```

```
tcp        0      0  *.telnet               *.*                    LISTEN
tcp        0      0  *.ftp                  *.*                    LISTEN
tcp4       0      0  hostrs6.32789          hostrs6.32790          ESTABLISHED
tcp4       0      0  hostrs6.32790          hostrs6.32789          ESTABLISHED
tcp4       0      0  *.32789                *.*                    LISTEN
tcp4       0      0  hostrs6.32781          hostrs6.918            ESTABLISHED
tcp4       0      0  hostrs6.918            hostrs6.32781          ESTABLISHED
tcp4       0      0  hostrs6.32787          hostrs6.32788          ESTABLISHED
tcp4       0      0  hostrs6.32788          hostrs6.32787          ESTABLISHED
tcp4       0      0  *.32787                *.*                    LISTEN
tcp4       0      0  hostrs6.32781          hostrs6.1004           ESTABLISHED
tcp4       0      0  hostrs6.1004           hostrs6.32781          ESTABLISHED
tcp4       0      0  hostrs6.32784          hostrs6.32785          ESTABLISHED
tcp4       0      0  hostrs6.32785          hostrs6.32784          ESTABLISHED
tcp4       0      0  *.32784                *.*                    LISTEN
tcp4       0      0  hostrs6.32781          hostrs6.808            ESTABLISHED
tcp4       0      0  hostrs6.808            hostrs6.32781          ESTABLISHED
tcp4       0      0  hostrs6.32782          hostrs6.32783          ESTABLISHED
tcp4       0      0  hostrs6.32783          hostrs6.32782          ESTABLISHED
tcp4       0      0  *.32782                *.*                    LISTEN
tcp4       0      0  hostrs6.32781          hostrs6.906            ESTABLISHED
tcp4       0      0  hostrs6.906            hostrs6.32781          ESTABLISHED
tcp4       0      0  *.32781                *.*                    LISTEN
tcp4       0      0  *.32780                *.*                    LISTEN
tcp4       0      0  loopback.smux          loopback.32779         ESTABLISHED
tcp4       0      0  loopback.32779         loopback.smux          ESTABLISHED
tcp4       0      0  *.smux                 *.*                    LISTEN
tcp4       0      0  *.smtp                 *.*                    LISTEN
tcp4       0      0  loopback.49213         *.*                    LISTEN
tcp4       0      0  *.writesrv             *.*                    LISTEN
tcp4       0      0  *.sunrpc               *.*                    LISTEN
tcp4       0      0  *.6000                 *.*                    LISTEN
tcp4       0      0  *.32768                *.*                    LISTEN
udp        0      0  *.tftp                 *.*
udp4       0      0  *.32780                *.*
udp4       0      0  *.time                 *.*
udp4       0      0  *.daytime              *.*
udp4       0      0  *.chargen              *.*
udp4       0      0  *.discard              *.*
udp4       0      0  *.echo                 *.*
udp4       0      0  *.32779                *.*
udp4       0      0  *.32778                *.*
udp4       0      0  *.32777                *.*
udp4       0      0  *.32776                *.*
udp4       0      0  *.32775                *.*
udp4       0      0  *.32774                *.*
udp4       0      0  *.ntalk                *.*
udp4       0      0  *.talk                 *.*
udp4       0      0  *.bootps               *.*
udp4       0      0  *.biff                 *.*
udp4       0      0  *.snmp                 *.*
udp4       0      0  *.32768                *.*
udp4       0      0  *.sunrpc               *.*
udp4       0      0  *.syslog               *.*
udp4       0      0  *.xdmcp                *.*
```

The best way to eliminate such vulnerabilities is to disable the services that start them.

A useful program to find out the programs that start these services is to use the lsof program. lsof is a GNU freeware and not part of AIX.

Here is a typical output from lsof:

```
# /usr/local/bin/lsof | grep -E "TCP|UDP|COMMAND"
lsof: WARNING: compiled for AIX version 4.3.2.0; this is 4.3.3.0.
COMMAND    PID USER   FD   TYPE     DEVICE   SIZE/OFF  NODE NAME
writesrv  2922 root    3u  IPv4 0x70073edc        0t0   TCP *:writesrv (LISTEN)
inetd     5418 root    4u  IPv4 0x700b36dc        0t0   TCP *:ftp (LISTEN)
inetd     5418 root    5u  IPv4 0x700b32dc        0t0   TCP *:telnet (LISTEN)
telnetd   8064 root    0u  IPv4 0x7012aedc      0t236   TCP host_a:telnet->leo:1345
(ESTABLISHED)
telnetd   8064 root    1u  IPv4 0x7012aedc      0t236   TCP host_a:telnet->leo:1345
(ESTABLISHED)
telnetd   8064 root    2u  IPv4 0x7012aedc      0t236   TCP host_a:telnet->leo:1345
(ESTABLISHED)
ftpd      8438 jane    0u  IPv4 0x700626dc      0t154   TCP host_a:ftp->host_b:33904
(ESTABLISHED)
ftpd      8438 jane    1u  IPv4 0x700626dc      0t154   TCP host_a:ftp->host_b:33904
(ESTABLISHED)
```

If you `grep` for TCP, and UDP, you will get a list of the programs on your machine that have the ports open. This will give you the PID and other useful information.

Looking at the /etc, we discover the startup scripts for the services:

- rc
- rc.C2
- rc.bsdnet
- rc.dacinet
- rc.ha_star
- rc.net
- rc.powerfail
- rc.tcpip
- rc.dt
- rc.net.serial
- rc.nfs

We rename the following dangerous services to prevent them from starting. Note that this is something that you should only do on your test machine.

```
# mv rc.dt Xrc.dt
# mv rc.net.serial Xrc.net.serial
# mv rc.nfs Xrc.nfs
```

Next, we drill inside rc.tcpip, which is responsible for starting the network daemons. In this case, we have disabled all daemons by commenting them out using `#`.

Options can also be added to daemons to provide additional security. The only daemon allowed to start is syslogd and we add a `-s` option to suppress logging for remote hosts:

```
# start /usr/sbin/syslogd "$src_running" -s
```

A selection of the rc.tcpip file is shown here:

```
# Start up dhcpcd daemon
#start /usr/sbin/dhcpcd "$src_running"

# Start up autoconf6 process
#start /usr/sbin/autoconf6 ""

# Start up ndpd-host daemon
#start /usr/sbin/ndpd-host "$src_running"

# Start up the ndpd-router daemon
#start /usr/sbin/ndpd-router "$src_running"

# Start up syslog daemon (for error and event logging)
start /usr/sbin/syslogd "$src_running" -s

# Start up print daemon
#start /usr/sbin/lpd "$src_running"

# Start up routing daemon (only start ONE)
#start /usr/sbin/routed "$src_running" -q
#start /usr/sbin/gated "$src_running"

# Start up the sendmail daemon.
#
# "/usr/lib/sendmail -bi" or "/usr/ucb/newaliases".
#

#qpi=30m  # 30 minute interval
#
#start /usr/lib/sendmail "$src_running" "-bd -q${qpi}"

# Start up Portmapper
#start /usr/sbin/portmap "$src_running"

# Start up socket-based daemons
#start /usr/sbin/inetd "$src_running"

# Start up Domain Name daemon
#start /usr/sbin/named "$src_running"
```

```
# Start up time daemon
#start /usr/sbin/timed "$src_running"

# Start up Network Time Protocol (NTP) daemon
#start /usr/sbin/xntpd "$src_running"

# Start up rwhod daemon (a time waster)
#start /usr/sbin/rwhod "$src_running"

# Start up the Simple Network Management Protocol (SNMP) daemon
#start /usr/sbin/snmpd "$src_running"

# Start up the DHCP Server
#start /usr/sbin/dhcpsd "$src_running"

# Start up the DHCP Relay Agent
#start /usr/sbin/dhcprd "$src_running"

# Start up the DPID2 daemon
#start /usr/sbin/dpid2 "$src_running"

# Start up the mrouted daemon
#start /usr/sbin/mrouted "$src_running"
```

If you are using inetd, there are several services that are still started after using securetcpip. In the following output, we show a sample of services commented off. We have left ftp and telnet uncommented as an example. But the rest of the chapter actually assumes that inetd is *not* started anyway:

```
#
ftp     stream  tcp6    nowait  root    /usr/sbin/ftpd         ftpd
telnet  stream  tcp6    nowait  root    /usr/sbin/telnetd      telnetd -a
#shell   stream  tcp6    nowait  root    /usr/sbin/rshd         rshd
#kshell  stream  tcp     nowait  root    /usr/sbin/krshd        krshd
#login   stream  tcp6    nowait  root    /usr/sbin/rlogind      rlogind
#klogin  stream  tcp     nowait  root    /usr/sbin/krlogind     krlogind
#exec    stream  tcp6    nowait  root    /usr/sbin/rexecd       rexecd
#comsat dgram   udp     wait    root    /usr/sbin/comsat       comsat
#uucp   stream  tcp     nowait  root    /usr/sbin/uucpd        uucpd
#bootpsdgram   udp     wait    root    /usr/sbin/bootpd       bootpd /etc/bootptab
##
## Finger, systat and netstat give out user information which may be
## valuable to potential "system crackers."  Many sites choose to disable
## some or all of these services to improve security.
##
#finger stream  tcp     nowait  nobody  /usr/sbin/fingerd     fingerd
#systatstreamtcpnowaitnobody/usr/bin/ps           ps -ef
#netstat streamtcpnowaitnobody/usr/bin/netstat      netstat -f inet
#
#tftp dgramudp6SRCnobody/usr/sbin/tftpdtftpd -n
#talk   dgram   udp     wait    root    /usr/sbin/talkd         talkd
#ntalk   dgram   udp     wait    root    /usr/sbin/talkd         talkd
#
```

```
# rexd uses very minimal authentication and many sites choose to disable
# this service to improve security.
#
#rquotad sunrpc_udp    udp    wait    root    /usr/sbin/rpc.rquotad rquotad 100011 1
#rexd sunrpc_tcptcpwaitroot/usr/sbin/rpc.rexd rexd 100017 1
#rstatd sunrpc_udpudpwaitroot/usr/sbin/rpc.rstatd rstatd 100001 1-3
#rusersd sunrpc_udpudpwaitroot/usr/lib/netsvc/rusers/rpc.rusersd rusersd 100002 1-2
#rwalld sunrpc_udpudpwaitroot/usr/lib/netsvc/rwall/rpc.rwalld rwalld 100008 1
#sprayd sunrpc_udpudpwaitroot/usr/lib/netsvc/spray/rpc.sprayd sprayd 100012 1
#pcnfsd sunrpc_udpudpwait root/usr/sbin/rpc.pcnfsd pcnfsd 150001 1-2
#echostreamtcpnowaitrootinternal
#discardstreamtcpnowaitrootinternal
#chargenstreamtcpnowaitrootinternal
#daytimestreamtcpnowaitrootinternal
#timestreamtcpnowaitrootinternal
#echodgramudpwaitrootinternal
#discarddgramudpwaitrootinternal
#chargendgramudpwaitrootinternal
#daytimedgramudpwaitrootinternal
#timedgramudpwaitrootinternal
## The following line is for installing over the network.
#instsrv streamtcpnowaitnetinst/u/netinst/bin/instsrv instsrv -r /tmp/netinstalllog
/u/netinst/scripts
#ttdbserversunrpc_tcptcpwaitroot/usr/dt/bin/rpc.ttdbserver rpc.ttdbserver 100083 1
#dtspcstreamtcpnowaitroot/usr/dt/bin/dtspcd /usr/dt/bin/dtspcd
#cmsdsunrpc_udpudpwaitroot/usr/dt/bin/rpc.cmsd cmsd 100068 2-5
#websmstreamtcpnowaitroot/usr/websm/bin/wsmserver wsmserver -start
```

Again, you comment out the services not needed using `#`.

Because we only need to allow specific services and deny all others, it's easier to comment out the entire file and uncomment the necessary services.

In the `ex` or `vi` command mode, you can easily comment the entire file using:

```
:1,$s/^#/
```

The final work needs to be done on inittab.

The following are started in the inittab (use rmitab to remove them):

poibe	Print back end
nfs	Network file system
writesrv	Write server (allows users to write back and forth)
pmd	Power management (do you really want your server to power off when it goes idle for a period of time?)
httpdlite	Lite NetQuestion Web server software

We can use `rmitab` to remove unnecessary services. For example, you can remove the writeserv service with the following command:

```
# rmitab writesrv
```

Note that removing nfs, poibe, and pmd may be considered unnecessary here because we have removed their startups previously. However, to be on the safe side, we use `rmitab` to remove them from inittab as follows:

```
init:2:initdefault:
brc::sysinit:/sbin/rc.boot 3 >/dev/console 2>&1 # Phase 3 of system boot
powerfail::powerfail:/etc/rc.powerfail 2>&1 | alog -tboot > /dev/console # Power
Failure Detection
rc:2:wait:/etc/rc 2>&1 | alog -tboot > /dev/console # Multi-User checks
fbcheck:2:wait:/usr/sbin/fbcheck 2>&1 | alog -tboot > /dev/console # run
/etc/firstboot
srcmstr:2:respawn:/usr/sbin/srcmstr # System Resource Controller
rctcpip:2:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
rcnfs:2:wait:/etc/rc.nfs > /dev/console 2>&1 # Start NFS Daemons
cron:2:respawn:/usr/sbin/cron
piobe:2:wait:/usr/lib/lpd/pio/etc/pioinit >/dev/null 2>&1  # pb cleanup
qdaemon:2:wait:/usr/bin/startsrc -sqdaemon
uprintfd:2:respawn:/usr/sbin/uprintfd
logsymp:2:once:/usr/lib/ras/logsymptom # for system dumps
pmd:2:wait:/usr/bin/pmd > /dev/console 2>&1 # Start PM daemon
diagd:2:once:/usr/lpp/diagnostics/bin/diagd >/dev/console 2>&1
dt:2:wait:/etc/rc.dt
cons:0123456789:respawn:/usr/sbin/getty /dev/console
```

At this stage, it is a good practice to reboot and confirm what you have done.

The final output of `netstat -a` should look like the following. Notice how all the services are gone except for syslog.

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
udp4       0      0  *.syslog               *.*

Active UNIX domain sockets
SADR/PCB  Type   Recv-Q Send-Q Inode    Conn     Refs     Nextref  Addr
70075c00 dgram       0      0 13811980        0        0        0 /dev/log
7007f2c0
70075a00 dgram       0      0 1370fc80        0        0        0
/dev/.SRC-unix/SRC0eed7a
7007f280
70075e00 dgram       0      0 1335e620        0        0        0 /dev/SRC
7007f300
700f4a00 dgram       0      0 13811ae0        0        0        0 /tmp/.PMDV1
700f0dc0
```

We have successfully removed every service besides syslog. This is only an example. You will, however, want to do something similar for your specific environment.

10.2.2 Removal of accounts

Remove the accounts for lpd, guest, uucp, and nuucp from your system using:

```
# rmuser -p <user>
```

Change the default shell to /bin/false for the following users:

Add `/bin/false` to the `usw` entry of /etc/security/login.cfg.

Change the users shell for daemon, bin, sys, adm, nobody:

```
# chsh <user> /bin/false
```

Again, this is a generic case and you have to make sure that the users required for your application are there.

10.2.3 NFS changes

If NFS is installed, simply comment off /etc/rc.nfs.

Update the following entries in /etc/rc.nfs:

```
dspmsg cmdnfs.cat -s 8 1 "NOT starting nfs services per \ LockDown:\n"
#LOCKDOWN# dspmsg cmdnfs.cat -s 8 1 "starting nfs service:\n"
#LOCKDOWN# if [ -x /usr/sbin/biod ] ; then
#LOCKDOWN#     start biod /usr/sbin/biod 8
#LOCKDOWN# fi

#LOCKDOWN# if [ -x /usr/sbin/nfsd -a -f /etc/exports ]; then
#LOCKDOWN# > /etc/xtab
#LOCKDOWN# /usr/sbin/exportfs -a
#LOCKDOWN# start nfsd /usr/sbin/nfsd 8
#LOCKDOWN# start rpc.mountd '/usr/sbin/rpc.mountd
#LOCKDOWN# fi
#LOCKDOWN#

#LOCKDOWN# if [ -x /usr/sbin/rpc.statd ]; then
#LOCKDOWN#  start rpc.statd /usr/sbin/rpc.statd
#LOCKDOWN# fi
#LOCKDOWN# if [ -x +/usr/sbin/rpc.lockd ]; then
#LOCKDOWN#  start rpc.lockd /usr/sbin/rpc.lockd
#LOCKDOWN# fi
```

To stop the current NFS, you may also issue:

```
# stopsrc -g nfs
```

10.2.4 Environment customization

This section runs through a checklist of items to be done for your environment:

- Update the Message of the Day (/etc/motd) with your security message. An example of this message may be as follows:

```
NOTICE TO USERS

Use of this machine waives all rights to your privacy,
and is consent to be monitored.
```

```
Unauthorized use prohibited.


login: "
```

- Update /etc/security/login.cfg. Enable SAK, and change the login herald:

```
sak_enabled=true
herald = "\r\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\ NOTICE TO
USERS\r\n\r\nUse of this machine waives all rights to your
privacy,\r\n\r\n and is consent to be monitored.\r\n\r\nUnauthorized
use prohibited.\r\n\r\n\r\nlogin: "
```

- Set default password restrictions in the /etc/security/user file, and enable SAK:

```
minage=0
maxage=12
maxexpired=4
minalpha=4
minother=1
minlen=6
mindiff=3
maxrepeats=3
histexpire=26
histsize=8
pwdwarntime=14
tpath=on
```

- Set root password restrictions in the /etc/security/user file:

```
maxage=5      #root's password must be changed after 5 weeks
minlen=8      #root's password must be 8 characters
rlogin=false  #root cannot remotely login
```

- Customize the user shell environment (/etc/profile, /etc/environment, /etc/security/.profile):
  - Remove the period from the PATH statement in /etc/profile, roots login script (usually /.profile), and /etc/environment.
  - Set the `TMOUT/TIMEOUT` variables in /etc/profile:

    ```
    TMOUT=600 ; TIMEOUT=600 ; export readonly TMOUT TIMEOUT
    ```

  - Set `EDITOR` for user in /etc/profile or /etc/environment:

    ```
    EDITOR=/usr/bin/vi
    ```

- Consider disabling login for root, so administrators must `su` to root. Add to the root's stanza in /etc/security/user:

```
login=false   #if you do not want root to login at the console
```

- Consider disabling all remote and dial-in terms at end of day, and enabling them in the morning, through /etc/security/login.cfg:

```
logintimes = 1-5:0730-1730
* Monday to Friday, 7:30 AM to 5:30 PM
```

- Change the order of host lookup:

```
echo "hosts=local4,bind4" > /etc/netsvc.conf
```

- Run tcbck and fix the problems:

```
tcbck -n tree
tcbck -t tree
```

The following is a list of files to back up before modifying:

- /.profile
- /etc/environment
- /etc/inetd.conf
- /etc/inittab -- use chitab or rmitab
- /etc/motd
- /etc/netsvc.conf
- /etc/profile
- /etc/rc.nfs
- /etc/rc.tcpip
- /etc/securetcpip
- /etc/security/login.cfg
- /etc/security/sysck.cfg
- /etc/security/user

10.3 Recommended day-to-day tasks

We recommend doing these activities on an on-going basis.

- Change root password on the first Monday of the month on all your systems.
- Create a bootable mksysb image of your system weekly. Keep your tapes for at least eight weeks.
- Maintain and enforce your security policy. Make sure all your users know what your security policy is and remind them quarterly of what their responsibilities are in protecting the assets.
- Monitor your log files:

```
/var/adm/sulog
/var/adm/wtmp
/etc/utmp
```

- Monitor your cron and at jobs:

```
cronadm at -l
cronadm root -l
```

- If you enabled auditing or accounting, monitor these files weekly.
- Run `tcbck -n tree` at least daily.

10.4 Regular system review

We recommend that you review your system for certain things on a scheduled basis. This will help you find security holes, and it will help you keep your system documented when changes happen. Regularly review the following:

- Run `tcbck -n tree` *manually* once a month so you see the output. Also, at this time, manually compare the sysck.cfg file with the backup on your write protected media with the `diff` command.
- Ensure that any security fixes are applied in a timely manner.
- Verify your LPPs (`lppchk`) once a month. This will show you other information about your files compared to the installed filesets.
- Verify your user configuration once a month. These commands verify consistency in the standard authentication methods.:

```
pwdck -n ALL
grpck -n ALL
usrck -n ALL
```

- Verify that the customizing that you did when you installed your computer is still in place. See 10.1, “Overview” on page 191.
- Run an internal security audit tool, such as tiger, to verify that you do not have a file or directory with insecure permissions. See Chapter 9, “Security products and software” on page 183 for more details.
- Run an external security audit tool, such as strobe, against your system to verify that you do not have any external security holes. See Chapter 9, “Security products and software” on page 183 for more details.

Appendix A. DoD classes

The “Orange Book” criteria came from a task force of the Defense Science Board started in 1967. The original report, “Security Controls for Computer Systems”, was published in 1970. The current document is “Trusted Computer System Evaluation” (DoD85). This material is important for many installations, but it is often misunderstood and misused.

There are two quite distinct sets of criteria. One set defines a number of security features. The other set defines the tools, information, and some of the processes required to verify the correctness (design and operation) of the features. The security features are the security policy and the second set is the assurance criteria.

Seven security levels are defined. These levels are:

- D - Minimal Protection
- C1 - Discretionary Security Protection
- C2 - Controlled Access Protection
- B1 - Labeled Security Protection, Mandatory Access Control
- B2 - Structured Protection
- B3 - Security Domains
- A1 - Verified Design

The following terms have specific meanings:

Security Policy
: These are the “rules” that the security features enforce. For example, every user must have a password, or only a file owner can change the access list for the file. The specific rules will vary in different security levels and in local installation standards. The total set of rules enforced by the system forms the security policy of the system.

Identification and Authentication (I&A)
: Subjects must be uniquely defined. A subject is a user or a process.

Marking or Labeling
: Objects (usually a file) must be associated with a security label that contains a security level and security category. For example, “Secret” is a level and “Research” might be a category.

Accountability

This refers to complete and secure records of actions that affect security. Such actions include user setup, assignment or change of security levels, and denied access attempts.

Assurance

This refers to system mechanisms that enforce security; it must be possible to measure the effectiveness of these mechanisms.

Continuous Protection

The hardware and software mechanisms that implement security must be protected against unauthorized change.

Object Reuse

This refers to memory blocks or disk blocks, for example. A program should not find “left over” data from another process or file when it acquires a memory or disk block.

Covert Channels

This refers to indirect means of delivering information to an unauthorized user. For example, a program might make subtle changes to unclassified messages to convey classified information, or leave data in a shared memory location.

Table 25 (taken from DoD85) contains the requirements for the various security classes.

Table 25. Requirements for the various security classes

Criteria	Classes						
	D	C1	C2	B1	B2	B3	A1
Security Policy							
Discretionary Access Control	x	R	R	-	-	R	-
Object Reuse	x	x	R	-	-	-	-
Labels	x	x	x	R	R	-	-
Label Integrity	x	x	x	R	-	-	-
Exportation of Labeled Information	x	x	x	R	-	-	-
Labeling Human-Readable Output	x	x	x	R	-	-	-
Mandatory Access Control	x	x	x	R	R	-	-
Subject Sensitivity Labels	x	x	x	x	R	-	-
Device Labels	x	x	x	x	R	-	-

Criteria	Classes						
	D	C1	C2	B1	B2	B3	A1
Accountability							
Identification and Authentication	x	R	R	R	-	-	-
Audit	x	x	R	R	R	R	-
Trusted Path	x	x	x	x	R	R	-
Assurance							
System Architecture	x	R	R	R	R	R	-
System Integrity	x	R	-	-	-	-	-
Security Testing	x	R	R	R	R	R	R
Design Specification/Verification	x	x	x	R	R	R	R
Covert Channel Analysis	x	x	x	x	R	R	R
Trust Facility Management	x	x	x	x	R	R	-
Trust Recovery	x	x	x	x	x	R	-
Trusted Distribution	x	x	x	x	x	x	R
Documentation							
Security Features User's Guide	x	R	-	-	-	-	-
Trusted Facility Manual	x	R	R	R	R	R	-
Test Documentation	x	R	-	-	R	-	R
Design Documentation	x	R	-	R	R	R	R
An "x" means no requirement. An "R" means this class has additional requirements over the lower classes. A "-" means this class has the same requirements as the next lower class.							

As can be seen, the requirements listed in Table 25 are very general. Many systems can claim to cover various levels of these requirements. To have any real meaning, a system must be certified for a particular level. This means that the system was examined (in great detail) by a U.S. Government agency and certified to operate at a certain security level. This certification is a long process and can be expensive for the system developer. The specific tests and criteria are designed for national security installations and may not be

completely appropriate for commercial users. Level “D” sometimes is applied to a system that failed tests for a higher level.

Certification does not provide a guarantee or warranty that the security system is perfect. It merely says that it satisfied the agency performing the tests. However, these tests are generally accepted to be rigorous.

Discretionary access control (DAC) and mandatory access control (MAC) are very important concepts. DAC allows the owner of a file to set the security parameters for the file. In AIX, this is the owner setting permission bits or ACL controls. MAC means that the system (through control parameters set by the security officer) automatically controls the security parameters of a file. The owner of the file cannot change these. AIX does not support MAC.

A system can have both MAC and DAC. The security officer (using various control lists) decides which files (or categories of files) are controlled through MAC and which are allowed for DAC. The DoD standard does not specify any particular implementation for these facilities, and different systems use very different mechanisms to implements these controls.

Human-readable output labels, specified in the DoD table, mean the system must automatically print security labels on output. This is independent of any particular application program. This can be a difficult requirement because the operating system does not understand what an application program is printing on any particular page. If the system overprints “TOP SECRET” at some page location, it may overlay important application output. (The MVS/RACF solution uses only laser page printers, and prints the security label in a nondestructive manner.)

A.1 Levels for commercial users

Class C is the most important security level for most commercial installations. (Almost all systems at this level control Object Reuse and provide an Audit facility, although these are C2 requirements only.) Class C is important for two key reasons, neither of which is directly related to the specific security features of C1 or C2:

1. A reasonable set of security features are included. This set is sufficient to provide reasonable control for most installations without creating a major administrative burden.
2. The system has been independently tested and certified to perform these functions correctly.

Unless there is a specific need for higher security, C1 or C2 should meet generally accepted security practice requirements—if used intelligently. From the user's point of view, there is little difference between C1 and C2.[1] The assurance and testing requirements for C2 are more rigorous and, in this sense, a C2 system is better than a C1 system.

From a normal commercial viewpoint, class B introduces two major changes: mandatory security access (MAC) and security labels. Security labels include both security level (secret, and so on) and category (research, payroll, and so on). A user might have access to secret data in research, but only unclassified data in payroll. The implementation of security labels requires substantial changes to most systems. Both security labels and MAC may require considerable administrative effort to implement and maintain.

The B2 and B3 classes become very rigorous and are difficult to certify. Class A security would be very unusual for a commercial installation.

A.2 Comments

System owners often assume the following:

- A higher security class is better.
- A higher class will take care of security needs with less effort.
- Certain applications need a higher security class.

These assumptions are all false and lead to misuse and a misunderstanding of the security classes.

One key factor is the sharing of systems. For example, consider a customer-account system in a bank. This appears to be a good candidate for a very high security level. However, if this system is implemented as a totally unshared system (that is, there are no users other than this application), and there are no "timesharing" terminals attached to the system, and physical security is sufficient, and so forth, then a formal security class is pointless. Likewise, a small departmental system shared by users of the same "security level" (whatever this may mean in a given organization) is unlikely to need as much protection as a large system shared by many varied users of different security levels and categories.

More security functions usually require more administrative effort. A higher class does not automatically provide more security. It is capable of providing more security with proper administration. Conversely, a higher class system may work very poorly and eventually become unusable if the security

[1] C2 requires an Auditing feature and control over Object Reuse. In practice, both of these are already in C1 systems.

functions are not properly administered. Do not buy or install more security than you are prepared to administer. AIX systems tend to be smaller than typical mainframe operating systems and may require a different viewpoint for administration and security. An AIX system is unlikely to have a formal security officer and probably does not have a full-time administrator of any kind.

AIX is designed to meet C2 security, and this document describes the administrative efforts necessary to install and maintain this class system.

The “Orange Book” specifications skip two especially important areas:

- Networking
- System updates

Most AIX systems are attached to local area networks, and it is sometimes difficult to clearly separate network security from individual system security. An individual system administrator has little control over general network security and must accept some network functions “on faith.” It is foolish, for example, to demand a B1 system and then connect it (with default, standard facilities) to an “open” TCP/IP network.[2]

Most commercially available operating systems have updates.[3] The DoD specifications seem to ignore these. It is not practical to re-certify a system for every update or “fix”. Thus, in principle, a system is uncertified after any update/fix is installed. The same concept applies when third-party software products having “authorized” modules are installed. In practice, these upgrades and products are accepted on faith.

Several members of the European community have produced their own information processing security criteria definition, named ITSEC. This started with the basic elements from the “Orange Book”, and then went in somewhat different directions. A discussion of ITSEC can be found in *The Library for System Solutions Security Reference*, GG24-4106.

[2] DoD defines a secure networking protocol named DODIIS Network Security for Information eXchange (DNSIX), which restricts import and export of classified data through network interfaces. Early versions of AIX 3 provided network interface controls matching parts of these functions. These functions were rarely used in “real-world” systems, and, in particular, were not used by generally-available TCP/IP functions. The TCP/IP community (as represented by the IETF, the Internet Engineering Task Force) has chosen other directions for future TCP/IP Security mechanisms.
[3] IBM sometimes calls these “PTF tapes” or “system maintenance tapes” or “system upgrades” and so forth.

Appendix B. Special notices

This publication is intended to help IBM customers, IBM business partners, and IBM I/T specialists concerned with AIX security. The information in this publication is not intended as the specification of any programming interfaces that are provided by AIX. See the PUBLICATIONS section of the IBM Programming Announcement for AIX for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no

guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AFP	AIX
AIXwindows	CICS
DB2	DPI
IBM	LoadLeveler
Micro Channel	Network Station
OS/2	RACF
RS/6000	S/390
SecureWay	SP
3890	

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli

A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix C. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 219.

- *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, SG24-5309
- *Elements of Security: AIX 4.1*, GG24-4433
- *The Library for System Solutions Security Reference*, GG24-4106 (Available in softcopy format only)

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

C.3 Other resources

These publications are also relevant as further information sources:

- *AIX Version 4.3 Commands Reference*, SBOF-1877
- *AIX Version 4.3 Files Reference*, SC23-4168
- *AIX Version 4.3 Installation Guide*, SC23-4112

- *AIX Version 4.3 Network Information Services (NIS and NIS+) Guide*, SC23-4310
- *AIX Version 4.3 System Management Concepts: Operating System and Devices*, SC23-4311
- *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127
- *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126
- *AIX Version 4.3 System User' s Guide: Communications and Networks*, SC23-4122 (Available online)
- *AIX Version 4.3 System User' s Guide: Operating System and Devices*, SC23-4121
- *AIX Version 4.3 Technical Reference: Communications Volume 1*, SC23-4161
- *AIX Version 4.3 Technical Reference: Communications Volume 2*, SC23-4162 (Available online)
- *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 1*, SC23-4159
- *Applied Cryptography*, written by Bruce Schneier, published by John Wiley & Sons, 1996
- *IBM RS/6000 Distributed System Trusted Facility Manual (TFM)*, (Available at: `http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/C2/tfm/toc.htm`)
- *TCP/IP Illustrated, Volume 1: The Protocols*, written by Richard Stevens, published by Addison-Wesley, 1994
- *The Cuckoo's Egg*, written by Clifford Stoll, published by Pocket Books, Reprinted 1995

C.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `ftp:// ftp.stanford.edu/general/security-tools/swatch`
- `ftp://ftp.cert.dfn.de/pub/tools/admin/Cops/`
- `ftp://ftp.cert.dfn.de/pub/tools/admin/L5/`
- `ftp://ftp.cert.dfn.de/pub/tools/admin/Tripwire/`
- `ftp://suburbia.net:/pub/stobe.tgz`

- http://rootshell.com
- http://techsupport.services.ibm.com/rs6k/techbrowse
- http://www.acm.org/classics/sep95
- http://www.bull.de/pub/out
- http://www.cert.dfn.de/eng/logsurf
- http://www.cert.org/ CERTR Coordination Center
- http://www.cert.org/advisories/CA-94.01.ongoing.network.monitoring.attacks.html
- http://www.clark.net/~roesch/security.html
- http://www.fish.com/~zen/satan/satan.html
- http://www.gocsi.com
- http://www.ietf.org/ The Internet Engineering Task Force
- http://www.ietf.org/html.charters/pkix-charter.html
- http://www.insecure.org/ Insecure.Org
- http://www.insecure.org/nmap
- http://www.iss.net/securing_e-business/security_products/security_assessment/system_scanner/index.php
- http://www.l0pht.com/users/10pht/nc110.tgz
- http://www.l0pht.com/users/10pht/nc11nt.zip
- http://www.nessus.org/ Nessus
- http://www.networkice.com
- http://www.nfr.com
- http://www.nist.gov/itl/lab/bulletns/dec99.htm
- http://www.pgp.com/asp_set/products/tns/ccscanner_intro.asp
- http://www.pwcglobal.com
- http://www.radium.ncsc.mil/tpep/library/tcsec/index.html
- http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/C2/tfm/toc.htm
- http://www.sans.org/mistakes.htm
- http://www.ssh.com
- http://www.ssh.org/ SSH Secure Shell
- http://www.starsecure.com

- http://www.tripwire.com
- http://www-4.ibm.com/software/network/directory/library/
- http://www-4.ibm.com/software/network/directory/library/publications/31/admin_help/parent.htm
- http://www-4.ibm.com/software/network/directory/library/publications/install_config/aparent.htm
- http://www-4.ibm.com/software/security/firewall

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Special symbols

". " (dot space) stealth directory. The ls command, without the -a argument, does not list file or directory names beginning with a dot. Even when listed, a dot-space directory may be unnoticed; and moreover, many users do not even know how to remove a dot-space directory should they notice its presence. For these reasons, hackers often store their software material in a dot-space directory somewhere on an attacked machine.

A

Access control list (ACL). A list of users and groups granted or denied given types of access on a given file, in addition to the standard UNIX (user, group, other) access mechanism. This is an AIX extension to UNIX to improve security.

Adapter. An adapter is a mechanism for attaching parts. For example, an adapter could be a part that electrically or physically connects a device to a computer or to another device. In the SP system, network connectivity is supplied by various adapters, some optional, that can provide connection to I/O devices, networks of workstations, and mainframe networks. Ethernet, FDDI, token ring, HiPPI, SCSI, SSA, FCS, and ATM are examples of adapters that can be used as part of an SP system.

Address. A character or group of characters that identifies a register, a device, a particular part of storage, or some other data source or destination.

AFS. A distributed file system that provides authentication services as part of its file system creation.

AIX. Abbreviation for Advanced Interactive Executive, IBMs licensed version of the UNIX operating system. AIX is particularly suited to support technical computing applications including high function graphics and floating point computations.

API. Application programming interface. A set of programming functions and routines that provide access between the application layer of the OSI seven-layer model and applications that want to use the network. It is a software interface.

Application. The use to which a data processing system is put, for example, a payroll application, an airline reservation application, and so on.

Application data. The data that is produced using an application program.

ARP (address resolution protocol). The mechanism used to translate logical IP addresses to physical MAC addresses.

Asymmetric encryption. An encryption scheme where different keys are used for encrypting and decrypting a message. In that way, the encryption key can be make public.

Authentication. The process of validating the identity of a user or server.

Authorization. The process of obtaining permission to perform specific actions.

B

Back door. A scheme allowing access to an account by supplying a predefined password instead of the real password.

Batch processing. (1) The processing of data or the accomplishment of jobs accumulated in advance in such a manner that each accumulation, thus formed, is processed or accomplished in the same run. (2) The processing of data accumulating over a period of time. (3) Loosely, the execution of computer programs serially. (4) Computer programs executed in the background.

BIND (Berkeley Internet Naming Daemon). The mechanism used to map dotted name addresses to IP addresses, and vice-versa.

BIOS (basic input/output system). The routines stored in non-volatile memory that provide base PC services.

Blackholing. Deliberately not giving an answer to a request, not even a rejection message. Blackholing makes the work of hackers more difficult.

bootp. The mechanism used to boot network devices that do not have non-volatile memory, or not enough. As it implies allowing the reading of files by something unknown on the network, it may be a security risk if improperly configured.

Breach. Any leak than can help a hacker to get information about the system, including those who can allow the hacker to penetrate the system.

Break attack. Some applications do not test interrupts and can be exited when their users just type break (Control-C), sometimes giving them access to a shell when they are not supposed to.

Broadcast. Sending a message to everything on the network. This is used by hackers for two purposes: (1) Making a list of hosts that answer to the broadcast so they can focus on attacking them; (2) Flooding the network with acknowledgment messages in order to organize denial of service.

Buffer overflow attack. Some applications ask for input without limiting the number of characters entered. If a hacker sends a long string and that this string overflow modifies some internal variables of the program, he/she may try to exploit that to modify the program behavior in order to get access to the system.

C

C2 security. One of the levels of security defined by the Department of Defense (DoD). AIX machines, both stand-alone and networked, can be defined to operate in C2 mode.

Challenge/answer. A scheme in which the password to be typed depends on information previously displayed by the system, either explicitly, or implicitly (date and time), and often a combination of both.

Challenge-Handshake Authentication Protocol (CHAP). A challenge/answer scheme in order to authenticate PPP communications.

Checksum. A number computed from the contents of a file in such manner that it is difficult to change the file contents without changing at the same time either this value or the file length.

CIFS (Computer Internet File System). The new name for PC communications formerly known as SMB.

Client. (1) A function that requests services from a server and makes them available to the user. (2) A term used in an environment to identify a machine that uses the resources of the network.

CMI. Centralized Management Interface. Provides a series of SMIT menus and dialogues used for defining and querying the SP system configuration.

Connectionless network. A network in which the sending logical node must have the address of the receiving logical node before information interchange can begin. The packet is routed through nodes in the network based on the destination address in the packet. The originating source does not receive an acknowledgment that the packet was received at the destination.

Consumable resources. The resources whose availability should be considered by LoadLeveler job scheduler to determine how to allocate tasks of a job on nodes. There are configuration-default consumable resources and also can be administrator-defined resources.

Control workstation. A single point of control allowing the administrator or operator to monitor and manage the SP system using the IBM AIX Parallel System Support Programs.

Crack. Test a long list of passwords against a list of encrypted passwords in order to find one that matches and get access to the system. In AIX, the list of encrypted passwords is not publicly readable as it was in former UNIXs, which virtually eliminates this risk.

Compiler. The main piece of software used to transform a human-readable program to binary code executable by the machine. Not having

compilers on a machines makes the hackers work difficult unless they bring their own with them.

Courtney. A publicly available program used to detect SATAN attacks on a system.

Credentials. Permissions accorded to a user, group, or host. Different credentials are managed at the AIX level, Secure NFS level, and NIS+ level.

Cryptography. The art of coding messages in order to make them secure (for system administrators); the art of trying to decode messages without having their key (for hackers).

CSS. Communication subsystem. Software that provides both user applications and kernel access to the switch adapter for communication and switch management purposes.

D

Daemon. A process, not associated with a particular user, that performs system-wide functions, such as administration and control of networks, execution of time-dependent activities, line printer spooling, and so forth.

DASD. Direct access storage device. Storage for input/output data.

Data encryption standard (DES). An encryption scheme used worldwide, originally devised by IBM, to ensure security of computer communications. The DES commonly uses a 56-bit keys.

Department of Defense (DoD). A computer user strongly concerned about security, which defined different computer security classes.

Denial of Service (DOS). Flooding a host, application, or display with pointless information so they will not be available to ensure their service.

DFS. Distributed File System. A subset of the IBM Distributed Computing Environment.

Digital signature. Encrypting a message with your private key before crypting it with the recipient's public key. Once the recipient receives the message and decodes it with his/her private key, if they can get something readable using your public key, it proves the message was coded with your private key; therefore, you can be considered as being its author.

DNS spoofing. Fooling the name server, so requests to given sites will be directed to a hacked site with an IP address different from the real one.

Domain. (1) In DNS, the name of a zone administrated by a master name server; most of the time, it is a qualified (=separated with dots) name. (2) In NIS+, the name of a zone administrated by an NIS+ master server; it *has* to be a qualified name.

Domain Name System (DNS). A way to resolve dotted names to IP addresses and vice versa. In UNIX, this is done using BIND, the Berkeley Internet Name Daemon.

DPCL. Dynamic Probe Class Library. A C++ class library whose API enables a program to dynamically insert probes into an executing program. DPCL can be used to create client/server-type analysis tools and also provides greater flexibility and inter-operability among such tools.

E

Encryption. Coding information so it is (in theory) impossible to read by someone who does not have the required decoding key.

Ethernet. (1) Ethernet is the standard hardware for TCP/IP local area networks in the UNIX marketplace. It is a 10-megabit per second baseband type LAN that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by collision detection (CSMA/CD). (2) A passive coaxial cable whose interconnections contain devices or components, or both, that are all active. It uses CSMA/CD technology to provide a best-effort delivery system.

Exploits. The way to use a security breach. Hackers exchange exploit information on the Internet.

F

Failed login. When a login is unsuccessful. AIX registers the date and time of that fact.

Failover. The assuming of server responsibilities by the node designated as backup server when the primary server fails.

Failure group. A collection of disks that share common access paths or adaptor connection and could all become unavailable through a single hardware failure.

Fake shell. A program that looks, from the hackers point of view, like a shell, but does not give any access to anything. Monitoring the hackers behavior on a fake shell can help in understanding what the hacker is looking for as well as his/her motivation in getting it. A good fake shell generate delays to every response in order to demotivate the hacker.

Fall back. Also called fallback, the sequence of events when a primary or server machine takes back control of its workload from a secondary or backup machine.

FFDC. First failure data capture. Facility aimed to enhance SP and cluster software serviceability by allowing applications to maintain a hierarchy of error messages.

Fiber distributed data interface (FDDI). An American National Standards Institute (ANSI) standard for 100-megabit-per-second LAN using optical fiber cables. An FDDI local area network (LAN) can be up to 100 km (62 miles) and can include up to 500 system units. There can be up to 2 km (1.24 miles) between system units and/or concentrators.

File transfer protocol (FTP). The Internet protocol (and program) used to transfer files between hosts. It is an application layer protocol in TCP/IP that uses Telnet and TCP protocols to transfer bulk-data files between machines or hosts.

File. A set of related records treated as a unit. For example, in stock control, a file could consist of a set of invoices.

File name. A CMS file identifier in the form of 'filename filetype filemode (such as TEXT DATA A).

File server. A centrally located computer that acts as a storehouse of data and applications for numerous users of a local area network.

Filtering. Automatically monitoring packets arriving on given ports in order to allow only those with given characteristics.

finger. A program, part of UNIX, designed to give information about a given user (name, office, service, phone, and so on). As such, one of the most useful social engineering tools for hackers.

Firewall. A host dedicated to be used as a boundary between a site (or a corporation) and the external world. It is generally used to transmit most requests coming from the inside and discard a number of requests coming from the outside.

Fragment. The space allocated an amount of data (usually the end of a file) too small to require a full block consisting of one or more subblocks (one thirty-second of block size).

Flooding. Sending packets at a rate such that the host, application, or adapter dealing with them cannot cope with their rate of arrival. This is generally done by having many hosts attacking a given victim simultaneously.

FTP. File transfer protocol, a means to exchange files between hosts.

G

Gateway. An intelligent electronic device interconnecting dissimilar networks and providing protocol conversion for network compatibility. A gateway provides transparent access to dissimilar networks for nodes on either network. It operates at the session presentation and application layers.

getty. The program waiting for a logon at every enabled port.

Group. A set of users having common access needs.

H

Hacker. A person trying to enter a system he/she is not registered on, whether for mere curiosity (standard hacking), to check system security in a

framework of a mission (ethical hacking), or for any other reason, such as spying or sabotage. However, most hackers refer to the third kind of people as crackers rather that hackers.

Hacking run. What happens between the time a hacker enters a system and leaves it. Analyzing hacking runs is an important aspect of computer security, generally performed by analyzing accounting and audit log files.

HACWS. High Availability Control Workstation function, based on HACMP, provides for a backup control workstation for the SP system.

Hashed Shared Disk (HSD). The data striping device for the IBM Virtual Shared Disk. The device driver lets application programs stripe data across physical disks in multiple IBM Virtual Shared Disks, thus reducing I/O bottlenecks.

Hello protocol. The routing protocol used to check the status between the NSFnet (National Science Foundation Network) nodes.

High availability cluster multiprocessing. An IBM facility to cluster nodes or components to provide high availability by eliminating single points of failure.

History file. The file listing the last commands issued by a user. As far as security is concerned, history files provide much information to hackers, but do not give any about them because they clean them before leaving.

Hole. A synonym for security breach.

Host. A computer connected to a network, providing an access method to that network. A host provides end-user services.

I

IBM Virtual Shared Disk. A subsystem that allows application programs executing on different nodes access to a raw logical volume as if it were local at each node.

IDEA. An 8-round cipher with a 64-bit block size, using 128-bits keys. IDEA is resistant to both differential and linear cryptanalysis. Currently, there is no known way of breaking IDEA short of brute force.

i-node. The internal structure that describes an individual file to AIX. An i-node contains file size and update information as well as the addresses of data blocks, or in the case of large files, indirect blocks that, in turn, point to data blocks. One i-node is required for each file.

Insider attack. An attack by a user who already has, legally or not, an account on the system. As in military operations, it is easier to attack a site once you have a foot in it.

Internet. A specific inter-network consisting of large national backbone networks, such as APARANET, MILNET, and NSFnet, and a myriad of regional and campus networks all over the world. The network uses the TCP/IP protocol suite.

Internet protocol (IP). (1) A protocol that routes data through a network or interconnected networks. IP acts as an interface between the higher logical layers and the physical network. This protocol, however, does not provide error recovery, flow control, or guarantee the reliability of the physical network. IP is a connectionless protocol. (2) A protocol used to route data from its source to it destination in an Internet environment.

Intranet. A network conforming to IP protocols and used by a corporation. It is generally protected from the rest of the Internet by a firewall.

IP address. A 32-bit address (in Version 4) assigned to devices or hosts in an IP Internet that maps to a physical address. The IP address is composed of a network and host portion. Version 6 of IP uses 128-bit addresses.

IP spoofing. Defining one's machine as having the IP address of another. It should not be confused with DNS spoofing.

J

Journaled file system. The local file system within a single instance of AIX.

K

Kamikaze packet. A packet sent to cause trouble on a network, sometimes because it has a size difficult to cope with,

Kerberos. A service for authenticating users in a network environment.

Kernel. The core portion of the UNIX operating system that controls the resources of the CPU and allocates them to the users. The kernel is memory-resident, is said to run in *kernel mode*, and is protected by the hardware from user tampering.

Keystroke monitoring. Capturing everything that is typed on a keyboard. This can be done at a short distance using electronic interference, but X-window, when improperly configured, presents the risk of allowing it from anywhere.

L

LAN. (1) Acronym for local area network, a data network located on the user's premises in which serial transmission is used for direct data communication among data stations. (2) Physical network technology that transfers data a high speed over short distances. (3) A network in which a set of devices is connected to another for communication and that can be connected to a larger network.

LAPI. Low Level Application Programming Interface. A non-standard IBM communication protocol which is designed to provide optimal communication performance on the SP switch network. The LAPI library provides PUT and GET functions and a general Active Message function to allow programmers to supply extensions by means of additions to the notification handlers.

Local host. The computer to which a user's terminal is directly connected.

Logical volume manager. Manages disk space at a logical level. It controls fixed-disk resources by mapping data between logical and physical storage allowing data to be discontinuous, span multiple disks, replicated, and dynamically expanded.

LWCF. Light Weight Core File. A non-standard AIX core file that only contains simple process stack traces. It doesn't have the low-level detail like in the traditional AIX core file. It is generally much smaller in size than traditional standard AIX core file.

M

MAC (media access content) address. A 6-byte number that is used to address a given adapter on a network.

MAN (metropolitan area network). A network to connect computers at distances where they cannot share the same LAN. MAN access is not compatible with a C2 network.

Metadata. Data structures that contain access information about file data. These might include i-nodes, indirect blocks, and directories. These data structures are used by GPFS but are not accessible to user applications.

Mirroring. The creation of a mirror image of data to be preserved in the event of disk failure.

MIT (Massachusetts Institute of Technology). The place where X-window was developed.

MPI. Message Passing Interface. An industry standard parallel programming interface that provides message passing libraries to parallelize a job by exchanging messages between more than two tasks. The latest release of the MPI standard is MPI 2.0, which is also referred to as MPI-2.

MPI hints. An MPI facility which provides information about things, such as the structure of the application, the type of expected file accesses, and preferences for node selection for running a set of tasks. The MPI standard defines the reserved hints and also allows for vendors' own implementation.

MPI-IO. The I/O component of MPI. It provides a set of interfaces to perform portable and efficient parallel I/O.

MPI One-sided Communication. MPI functionality that extends the communication mechanisms of MPI by allowing one process to specify all communication parameters, both for the sending and receiving sides of message passing. It is also referred to as MPI 1-sided.

MSS. Master Switch Sequencing node. A node that periodically re-sequences the TOD signals on the SP Switch2.

N

Network. An interconnected group of nodes, lines, and terminals. A network provides the ability to transmit data to and receive data from other systems and users.

NFS. Network file system. NFS allows different systems (UNIX or non-UNIX), different architectures, or vendors connected to the same network to access remote files in a LAN environment as though they were local files. Version 3 of NFS uses TCP. Previous versions could only use UDP.

NIS+ (Network Information System Plus). A way to give users access to many machines while still keeping the same password on all of them. NIS was another way (less secure, and now obsolete) to do the same thing.

NOLOGIN. A way to ensure that nobody can log on directly as a given user. On C2 systems, the root account is defined as NOLOGIN.

O

ODM (object data manager). In AIX, a hierarchical object-oriented database for configuration data.

One-time passwords. Passwords that are used only once, so sniffing them on a network is of no use.

P

Packet filter. A way to let some packets go through something while blocking others.

Packet sniffing. See sniffing.

Parallel environment. A system environment where message passing or SP resource manager services are used by the application.

Parallel Environment. A licensed IBM program used for message passing applications on the SP or RS/6000 platforms.

Parallel processing. A multiprocessor architecture that allows processes to be allocated to tightly coupled multiple processors in a cooperative processing environment allowing concurrent execution of tasks.

Parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) An item in a menu for which the operator specifies a value or for which the system provides a value when the menu is interpreted. (3) A name in a procedure that is used to refer to an argument that is passed to the procedure. (4) A particular piece of information that a system or application program needs to process a request.

Password history information. A set of previous passwords, remembered by the system, so a user cannot use the same password too often.

Path. The ordered list of directories searched to execute a command. By modifying the path (contained in environment variable PATH), a user can build a Trojan Horse.

Primary node or machine. (1) A device that runs a workload and has a standby device ready to assume the primary workload if that primary node fails or is taken out of service. (2) A node on the SP Switch that initializes, provides diagnosis and recovery services, and performs other operations to the switch network. (3) In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk, and the other is designated the secondary, or backup, node. The primary node is the server node for IBM Virtual Shared Disks defined on the physical disks under normal conditions. The secondary node can become the server node for the disks if the primary node is unavailable (offline or down).

Primary server. When physical disks are connected to two nodes (twin-tailed), this is the node that normally maintains and controls local access to the disk.

Private key. The key used to decrypt information coded using one's public key. Nobody should know a given private key except the owner of this private key.

Process. (1) A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. (2) Any operation or combination of operations on data. (3) A function being performed or waiting to be performed. (4) A program in operation. For example, a daemon is a system process that is always running on the system.

Protocol. A set of semantic and syntactic rules that defines the behavior of functional units in achieving communication.

Public key. A publicly advertised key that can be used by anybody to send things to the owner of the corresponding private key.

Q

Quorum. The minimum number of nodes that must be running in order for the GPFS daemon to start. This is one plus half of the number of nodes in the GPFS configuration.

Quota. The amount of disk space and number of i-nodes assigned as upper limits for a specified user or group of users.

R

RAID. Redundant Array of Independent Disks. A set of physical disks that act as a single physical volume and use parity checking to protect against disk failure.

RARP (Reverse Address Resolution Protocol). A way to know the IP address of a device knowing its MAC address.

Recovery. The process of restoring access to file system data when a failure has occurred. This may involve reconstructing data or providing alternative routing through a different server.

Replication. The practice of creating and maintaining multiple file copies to ensure availability in the event of hardware failure.

Reverse engineering. Analyzing a program in an attempt to understand it and rebuild its source.

RISC. Reduced instruction set computing. The technology for today's high-performance personal computers and workstations, was invented in 1975. Uses a small, simplified set of frequently used instructions for rapid execution.

rlogin (remote LOGIN). A service offered by Berkeley UNIX systems that allows authorized users of one machine to connect to other UNIX systems across a network and interact as if their terminals were connected directly. The rlogin software passes information about the user's environment (for example, terminal type) to the remote machine.

RPC. Acronym for remote procedure call, a facility that a client uses to have a server execute a procedure call. This facility is composed of a library of procedures plus an XDR.

RSA (Rivest-Shamir-Adelman). The three authors of a powerful crypting scheme designed by the same initials.

RSH. A variant of the RLOGIN command that invokes a command interpreter on a remote UNIX machine and passes the command line arguments to the command interpreter, thus, skipping the LOGIN step completely. See also rlogin.

S

SATAN (System Administrators Tool for Analyzing Networks). A freeware package allowing system administrators to probe their systems and test their vulnerabilities. Also used remotely by hackers looking for sites with breaches allowing to hack them. Also named SANTA.

SAK (in TCB).

SCSI. Small computer systems interface. An adapter supporting attachment of various direct-access storage devices.

SDR. System Data Repository. Database for SP system configuration information. Resides on the control workstation only. Information entered through the SMIT panels or commands during installation goes into the SDR. Much of the information in the SDR can be viewed and some can be entered or changed using SP Perspectives or PSSP commands.

Secondary node. In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk and the other is designated as the secondary, or backup, node. The secondary node acts as the server node for the IBM Virtual Shared disks defined on the physical disks if the primary node is unavailable (offline or down).

Secondary server. The second node connected to a twin-tailed disk. This node assumes control of local access if the primary server fails.

Server. (1) A function that provides services for users. A machine may run client and server processes at the same time. (2) A machine that provides resources to the network. It provides a network service, such as disk storage and file transfer, or a program that uses such a service. (3) A device, program, or code module on a network dedicated to providing a specific service to a network. (4) On a LAN, a data station that provides facilities to other data stations. Examples are file server, print server, and mail server.

Shared Memory MPI. An implementation of MPI that allows tasks of a parallel job on the same node to send or receive their messages through the system's shared memory instead of physical network. PE 3.2 supports the shared memory MPI.

Shell. The shell is the primary user interface for the UNIX operating system. It serves as command language interpreter, programming language, and allows foreground and background processing. There are three different implementations of the shell concept: Bourne, C, and Korn.

SMIT. The System Management Interface Tool is a set of menu driven utilities for AIX that provides functions, such as transaction login, shell script creation, automatic updates of object database, and so forth.

SNMP. Simple Network Management Protocol. (1) An IP network management protocol that is used to monitor attached networks and routers. (2) A TCP/IP-based protocol for exchanging network management information and outlining the structure for communications among network devices.

Socket. (1) An abstraction used by Berkeley UNIX that allows an application to access TCP/IP protocol functions. (2) An IP address and port number pairing. (3) In TCP/IP, the Internet address of the host computer on which the application runs and the port number it uses. A TCP/IP application is identified by its socket.

SSA. Serial storage architecture. An expanded storage adapter for multiprocessor data sharing in UNIX-based computing allowing disk connection in a high-speed loop.

Standby node or machine. A device that waits for a failure of a primary node in order to assume the identity of the primary node. The standby machine then runs the primary's workload until the primary is back in service.

Stripe group. A file system written across many disks, which are connected to multiple nodes.

Striping. A method of writing a file system, in parallel, to multiple disks instead of to single disks in a serial operation.

Sub-block. The smallest unit of data accessible in an I/O operation equal to one thirty-second of a data block.

Subnet. Shortened form of subnetwork.

Subnet Mask. A bit template that identifies to the TCP/IP protocol code the bits of the host address that are to be used for routing for specific subnetworks.

Subnetwork. Any group of nodes that have a set of common characteristics, such as the same network ID.

Subsystem. A software component that is not usually associated with a user command. It is usually a daemon process. A subsystem will perform work or provide services on behalf of a user request or operating system request.

Switch. A message-passing network that connects all processor nodes with a minimum of four paths between every pair of nodes.

T

tar. Tape archive is a standard UNIX data archive utility for storing data on tape media.

TCP. Acronym for Transmission Control Protocol, a stream communication protocol that includes error recovery and flow control.

TCP/IP. Acronym for Transmission Control Protocol/Internet Protocol, a suite of protocols designed to allow communication between networks regardless of the technologies implemented in each network. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It assumes that the underlying protocol is the Internet Protocol.

Telnet. Terminal Emulation Protocol, a TCP/IP application protocol that allows interactive access to foreign hosts.

Token management. A system for controlling file access in which each application performing a read or write operation is granted exclusive access to a specific block of file data. This ensures data consistency and controls conflicts.

Token ring. (1) Network technology that controls media access by passing a token (special packet or frame) between media-attached machines. (2) A network with a ring topology that passes tokens from one attaching device (node) to another. (3) The IBM Token-Ring LAN connection allows the RS/6000 system unit to participate in a LAN adhering to the IEEE 802.5 Token Passing Ring standard or the ECMA standard 89 for Token Ring, baseband LANs.

Transaction. An exchange between the user and the system. Each activity the system performs for the user is considered a transaction.

U

UNIX Operating System. An operating system developed by Bell Laboratories that features multiprogramming in a multiuser environment. The UNIX operating system was originally developed for use on minicomputers but has been adapted for mainframes and microcomputers. Note: The AIX operating system is IBMs implementation of the UNIX operating system.

User. Anyone who requires the services of a computing system.

User datagram protocol (UDP). (1) In TCP/IP, a packet-level protocol built directly on the Internet Protocol layer. UDP is used for application-to-application programs between TCP/IP host systems. (2) A transport protocol in the Internet suite of protocols that provides unreliable, connectionless datagram service. (3) The Internet Protocol that enables an application programmer on one machine or process to send a datagram to an application program on another machine or process.

User ID. A non-negative integer, contained in an object of type uid_t, that is used to uniquely identify a system user.

V

Virtual Shared Disk, IBM. The function that allows application programs executing at different nodes of a system partition to access a raw logical volume as if it were local at each of the nodes. In actuality, the logical volume is local at only one of the nodes (the server node).

W

Workstation. (1) A configuration of input/output equipment at which an operator works. (2) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

X

X-Windows system. A graphical user interface product.

Abbreviations and acronyms

ABI	Application Binary Interface
ACL	Access Control List
ADP	Application Development Package
AFPA	Adaptive Fast Path Architecture
AFS	Andrews File System
AH	Authentication Header
ANSI	American National Standards Institute
API	Application Programming Interface
ARP	Address Resolution Protocol
ASR	Address Space Register
ATM	Asynchronous Transfer Mode
AUI	Attached Unit Interface
AWT	Abstract Window Toolkit
BIND	Berkeley Internet Name Daemon
BOS	Base Operating System
BLOB	Binary Large Object
BNU	Basic Network Utilities
BSC	Binary Synchronous Communications
CA	Certificate Authority
CDE	Common Desktop Environment
CDLI	Common Data Link Interface
CD-R	CD Recordable
CDSA	Common Data Security Architecture
CE	Customer Engineer
CEC	Central Electronics Complex
CGE	Common Graphics Environment
CHRP	Common Hardware Reference Platform
CISPR	International Special Committee on Radio Interference
CLVM	Concurrent LVM
CMOS	Complimentary Metal-Oxide Semiconductor
COFF	Common Object File Format
CORBA	Common Object Request Broker
CSID	Character Set ID
CSSM	Common Security Services Manager
DAC	Discretionary Access Control
DAD	Duplicate Address Detection
DASD	Direct Access Storage Device
DBE	Double Buffer Extension
DBCS	Double Byte Character Set
DCE	Distributed Computing Environment
DCN	Distributed Computer Network

DDOS	Distributed Denial-of-Service
DES	Data Encryption Standard
DFS	Distributed File System
DHCP	Dynamic Host Configuration Protocol
DIT	Directory Information Tree
DMA	Direct Memory Access
DMT	Directory Management Tool
DN	Distinguished Name
DNS	Domain Name System
DOS	Denial-of-Service
DS	Differentiated Service
DSE	Diagnostic System Exerciser
DSMIT	Distributed SMIT
DSSS	Direct Sequence Spread Spectrum
DTE	Data Terminating Equipment
DUA	Directory User Agent
EA	Effective Address
ECC	Error Checking and Correcting
EGP	Exterior Gateway Protocol
EIA	Electronic Industries Association
EMU	European Monetary Union
EOF	End of File
ESID	Effective Segment ID
ESP	Encapsulating Security Payload
FCAL	Fibre Channel Arbitrated Loop
FCC	Federal Communication Commission
FDDI	Fiber Distributed Data Interface
FDPR	Feedback Directed Program Restructuring
FHSS	Frequency Hopping Spread Spectrum
FIFO	First In/First Out
FLASH EPROM	Flash Erasable Programmable Read-Only Memory
FLIH	First Level Interrupt Handler
FRCA	Fast Response Cache Architecture
FTP	File Transfer Protocol
GAI	Graphic Adapter Interface
GID	Group ID
GPR	General Purpose Register
GSKit	Global Security Kit
GUI	Graphical User Interface
HACMP	High Availability Cluster Multi-Processing
HCON	IBM AIX Host Connection Program/6000
HFT	High Function Terminal
HTTP	Hypertext Transfer Protocol
IAR	Instruction Address Register
ICCCM	Inter-Client Communications Conventions Manual

ICE Inter-Client Exchange

ICElib Inter-Client Exchange Library

ICMP Internet Control Message Protocol

IDS Intrusion Detection Systems

IETF Internet Engineering Task Force

IFS Internal Field Separator

IHV Independent Hardware Vendor

IIOP Internet Inter-ORB Protocol

IJG Independent JPEG Group

IKE Internet Key Exchange

ILS International Language Support

IM Input Method

IMAP Internet Message Protocol

INRIA Institut National de Recherche en Informatique et en Automatique

IPL Initial Program Load

IPSec IP Security

IS Integrated Service

ISA Industry Standard Architecture

ISAKMP/Oakley Internet Security Association Management Protocol

ISNO Interface Specific Network Options

ISO International Organization for Standardization

ISV Independent Software Vendor

IT Information Technology

ITSO International Technical Support Organization

I/O Input/Output

JDBC Java Database Connectivity

JFC Java Foundation Classes

JFS Journaled File System

JNDI Java Naming and Directory Interface

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

LDIF LDAP Directory Interchange Format

LFT Low Function Terminal

LID Load ID

LP Logical Partition

LPD Line Printer Daemon

LPI Lines Per Inch

LPP Licensed Program Products

LPR/LPD Line Printer/Line Printer Daemon

LP64 Long-Pointer 64

LRU Least Recently Used

LTG Logical Track Group

LV Logical Volume

LVCB Logical Volume Control Block

LVD Low Voltage Differential

LVM Logical Volume Manager

L2 Level 2

MAC Mandatory Access Control

MBCS MultiByte Character Support

MCA Micro Channel Architecture

MDI Media Dependent Interface

MII Media Independent Interface

MIS Management Information System

MODS Memory Overlay Detection Subsystem

MP Multiple Processor

MPOA Multiprotocol Over ATM

MST Machine State

NBC Network Buffer Cache

ND Neighbor Discovery

NDP Neighbor Discovery Protocol

NFS Network File System

NHRP Next Hop Resolution Protocol

NIM Network Installation Management

NIS Network Information System

NIST National Institute of Standards and Technology

NL National Language

NLS National Language Support

NTF No Trouble Found

NVRAM Non-Volatile Random Access Memory

OACK Option Acknowledgment

ODBC Open DataBase Connectivity

ODM Object Data Manager

OEM Original Equipment Manufacturer

OLTP Online Transaction Processing

ONC+ Open Network Computing

OOUI Object-Oriented User Interface

OSF Open Software Foundation, Inc.

PCI Peripheral Component Interconnect

PDT Paging Device Table

PEX PHIGS Extension to X

PFS Perfect Forward Security

PHB Processor Host Bridges

PHY Physical Layer Device

PID Process ID

PII Program Integrated Information

PMTU Path MTU

POP Power-On Password

PPC PowerPC

PPP Point-to-Point Protocol

PSE Portable Streams Environment

PTF Program Temporary Fix

PTS Pseudo Terminal Slave

PV Physical Volume

QoS Quality of Service

RAID Redundant Array of Independent Disks

RAN Remote Asynchronous Node

RAS Reliability Availability Serviceability

RDB Relational DataBase

RDISC ICMP Router Discovery

RDN Relative Distinguished Name

RDP Router Discovery Protocol

RFC Request for Comments

RIO Remote I/O

RIP Routing Information Protocol

RPA RS/6000 Platform Architecture

RPC Remote Procedure Call

RPL Remote Program Loader

RSVP Resource Reservation Protocol

SA Secure Association

SACK Selective Acknowledgments

SAK Secure Attention Key

SAN Storage Area Network

SASL Simple Authentication and Security Layer

SBCS Single-Byte Character Support

SCB Segment Control Block

SCSI Small Computer System Interface

SCSI-SE SCSI-Single Ended

SDRAM Synchronous DRAM

SE Single Ended

SET Secure Electronic Transfer

SGID Set Group ID

SHLAP Shared Library Assistant Process

SID Segment ID

SIT Simple Internet Transition

SKIP Simple Key Management for IP

SLB Segment Lookaside Buffer

SLIH Second Level Interrupt Handler

SLIP Serial Line Internet Protocol

SM Session Management

SMIT System Management Interface Tool

SMB Server Message Block

SMP Symmetric Multiprocessor

SMS System Management Services

SMTP Simple Mail Transfer Program

SNG Secured Network Gateway

SP Service Processor

SPCN System Power Control Network

SPI Security Parameter Index

SPM System Performance Measurement

SPOT Shared Product Object Tree

SRC System Resource Controller

SRN Service Request Number

SSA	Serial Storage Architecture
SSL	Secure Sockets Layer
STP	Shielded Twisted Pair
SUID	Set User ID
SVC	Supervisor or System Call
SVTX	Save Text
SYNC	Synchronization
TCB	Trusted Computing Base
TCE	Translate Control Entry
TCP/IP	Transmission Control Protocol/Internet Protocol
TCSEC	Trusted Computer System Evaluation Criteria
TFTP	Trivial File Transfer Protocol
TOS	Type Of Service
TTL	Time To Live
UCS	Universal Coded Character Set
UID	User ID
UIL	User Interface Language
ULS	Universal Language Support
UP	Uni-Processor
USLA	User-Space Loader Assistant
UTF	UCS Transformation Format
UTM	Uniform Transfer Model
UTP	Unshielded Twisted Pair
UUCP	UNIX-to-UNIX Copy Program
VFB	Virtual Frame Buffer
VFS	Virtual File System
VG	Volume Group
VGDA	Volume Group Descriptor Area
VGSA	Volume Group Status Area
VHDCI	Very High Density Cable Interconnect
VMM	Virtual Memory Manager
VP	Virtual Processor
VPD	Vital Product Data
VPN	Virtual Private Network
VSM	Visual System Manage
WAP	Wireless Application Protocol
WebSM	Web-based System Manager
WLM	Workload Manager
WTLS	Wireless Transport Layer Security
XCOFF	Extended Common Object File Format
XIE	X Image Extension
XIM	X Input Method
XKB	X Keyboard Extension
XOM	X Output Method
XPM	X Pixmap
XVFB	X Virtual Frame Buffer

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5962-00 AIX 4.3 Elements of Security Effective and Efficient Implementation
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

AIX 4.3 Elements of Security Effective and Efficient Implementation

AIX 4.3 Elements of Security

Effective and Efficient Implementation

Achieve confidentiality, integrity, and availability

Practical examples and best practice recommendations

Gain insight into AIX security management

This IBM Redbook provides an overview of AIX Version 4.3 security. AIX provides many security features that can be used to improve security. The emphasis is on the practical use of these security features, why they are necessary, and how they can be used in your environment. Also recommended are guidelines and best practices when there are many different ways to achieve a secure system.

The exponential growth of networks has caused more and more computers to be connected together, which creates an excellent environment for information exchange and sharing. As an increasingly large amount of confidential information is stored and transmitted over public networks, such as the Internet, it becomes imperative that information security be implemented in an effective and efficient manner. This book covers different aspects of security present in AIX 4.3, including user accounts, file systems, networks, and security management.

With its detailed product coverage, this book is intended for experienced AIX system administrators who are taking on the role of security administration. Security administrators who are new to AIX will also find this document useful. The reader is assumed to have a basic working knowledge of UNIX. This book is intended as an additional source of security

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-5962-00 ISBN 0738417149